UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Dpto. de INGENIERÍA DE SISTEMAS Y AUTOMÁTICA


INGENIERÍA TÉCNICA INDUSTRIAL
ESPECIALIDAD ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

# CAMINATA DEL ROBOT HUMANOIDE RH-2 EN LA PLATAFORMA DE SIMULACIÓN OPENHRP

**AUTOR**: CARLOS DE TORRE DOBLAS

**TUTOR**: CONCEPCIÓN ALICIA MONJE MICHARET

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Dpto. de INGENIERÍA DE SISTEMAS Y AUTOMÁTICA


INGENIERÍA TÉCNICA INDUSTRIAL
ESPECIALIDAD ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

# CAMINATA DEL ROBOT HUMANOIDE RH-2 EN LA PLATAFORMA DE SIMULACIÓN OPENHRP

**AUTOR**: CARLOS DE TORRE DOBLAS

**TUTOR**: CONCEPCIÓN ALICIA MONJE MICHARET

*A mis padres y amigos.*

# Agradecimientos

Me resulta emocionante escribir este texto porque significan el final de una etapa.
He de expresar mi profundo agradecimiento a todas aquellas personas que me han dado la oportunidad de desarrollarme, tanto intelectual como personalmente.
Para mí este trabajo significa la continuación de mi gran proyecto de vida y de las innumerables experiencias que me han hecho crecer aceleradamente.
Antes que nada se lo dedico a mis padres; por ser mis padres, por ser las personas con las que he contado, cuento y contaré incondicionalmente durante toda mi vida; sé que estarán siempre ahí, tanto en los malos como en los buenos momentos. A mi madre, que sin su presión, su hincapié y sus ánimos hubiera estudiado la mitad, gracias por entenderme y ser tan comprensiva conmigo. A mi padre, por ser tan especial, tan diferente al resto, por darme aire cuando lo necesito, porque sin su parte de restar importancia a las cosas, todo hubiera sido más difícil, gracias por esa libertad que me das. Porque ha sido un trabajo mío, pero con una gran recompensa que sé que ellos sentirán, me refiero a esa gran satisfacción personal (sobre todo mi madre).
A una persona no menos importante, mi hermana, ese ejemplo permanente que he tenido toda la vida, esa inteligencia con piernas que siempre me ha asombrado, esa persona tan perfecta. Gracias por prestarme tu ayuda cuando la he necesitado.
A todos mis profesores, desde el colegio hasta la universidad, en especial a mi profesor Donato, por ser el profesor que más tizas de colores puede llegar a utilizar para explicar los planos y vectores. Y Concha, mi tutora de este trabajo, por la gran paciencia que ha tenido. Por todo lo que he aprendido gracias a vosotros.
A todos mis compañeros y amigos de la universidad, sobre todo a mi compañero de proyecto Rubén, con el cual he seguido un camino a la par, siendo uña y carne. Porque sin todos vosotros, vuestros resúmenes, vuestros consejos, y vuestra ayuda seguro que no estaba escribiendo estas líneas.
A ella, como me iba a olvidar de ti, has estado en mi vida desde los 11 años, gracias por confiar siempre en mí y en mis capacidades. Gracias por querer estar en mi vida, y gracias por todas las cosas tan buenas que me aportas.

Carlos de Torre Doblas
22 de julio de 2010

# INDICE

ÍNDICE DE FIGURAS

# 1 Introducción

## 1.1 Introducción

Hoy en día la simulación de elementos mecánicos se hace imprescindible debido a la complejidad de los prototipos diseñados, así como al coste al que se arriesgan los creadores; por ello, el ver en un ambiente de realidad virtual lo que sucedería con ciertos elementos, variándolos y haciendo infinidad de pruebas, ahorra muchos recursos y cantidad de tiempo.

Esta herramienta permite, además de observar el movimiento de los mecanismos, saber sus fuerzas internas, sus pares, e infinidad de datos facilitados por sensores, los cuales sirven para saber, finalmente, qué elementos debemos utilizar para nuestro modelo.

Actualmente, existen herramientas de simulación virtual ajustadas al nivel del usuario, con una interfaz entendible y previsible, la cual facilita el trabajo con el software y no es necesario dominar lenguajes informáticos ni conocimientos avanzados.

## 1.2 Objetivos

En primer lugar, en este proyecto se trabajará con el simulador OpenHRP3, intentando comprender su funcionamiento interno (carga de archivos, lenguajes, funcionalidades, etc.). También se adaptará el modelo del robot humanoide Rh-2 al entorno del mismo, pudiendo así realizar simulaciones.

La verdadera finalidad de este proyecto, y en lo que se ha hecho especial incapié, es simular el robot sosteniendo una caja, validando así las fuerzas que sienten las distintas partes del cuerpo en los momentos críticos.
Para ello, se adaptan las medidas del Rh-2 y todos sus componentes en el simulador, después los archivos de velocidad y aceleración de cada una de las partes del robot y, finalmente, el peso de la caja para comprobar las variaciones de fuerza.
Los datos de fuerza observados en las distintas partes del robot se obtienen con sensores localizados en distintas articulaciones, dando así, las gráficas tiempo/par.

Lo primero a estudiar será el robot sin caja, dando un paso normal y viendo los pares que sienten las articulaciones de los hombros, codos, muñecas y tobillos. Esto permite la regulación del robot.

Posteriormente, será añadida una caja sobre los brazos, estudiando los de nuevo los pares que sienten dichas articulaciones. En el caso de los brazos, se estudiará un lado del robot, en concreto el izquierdo, ya que la diferencia en el paso es pequeña. Los tobillos se estudiarán los dos, y al final de la memoria, puesto que en el paso existen diferencias importantes entre ambos.

Se trabajarán dos ejemplos con cajas de distinta masa, siendo éstas de 3 Kg y 30 Kg. Con esto se pretende demostrar la fiabilidad del simulador, observando que la diferencia de par entre los dos casos es del orden de diez.

El estudio de los pares se hará respecto a los tres ejes de coordenadas: “x”, “y” y “z”, siendo el eje “y” donde más fuerzas siente el robot debido al peso de la caja.

## 1.3 Estructura del documento

El capítulo 1 "Introducción" comienza con una breve iniciación al proyecto y nos sitúa en el entorno en el que se va a desarrollar. Además, se exponen los objetivos principales y la estructura del mismo.

En el capítulo 2 "Descripción del simulador OpenHRP3" se trata todo lo referente a la plataforma OpenHRP3. Se explican los programas que son necesarios para el funcionamiento del simulador así como la estructura interna del simulador (servidores) y su interfaz gráfica.

En el capítulo 3 "Modelo de la plataforma RH-2 en OpenHRP3" se resume de forma breve el modelado del robot en el lenguaje VRML, dado que mi compañero lo ha explicado íntegramente en su proyecto.

El capítulo 4 "Simulación de tareas en OpenHRP3" contiene la información de los archivos necesarios para la simulación de tareas en la plataforma OpenHRP3 y los pasos a seguir para realizarla.

El capítulo 5 "Casos prácticos del robot RH-2 en OpenHRP3" explica las tareas del robot RH-2 simuladas en OpenHRP3. Además, se muestran los datos obtenidos de la simulación.

Las conclusiones a las que se ha llegado se muestran en el capítulo 6. Mientras que los posibles trabajos futuros y líneas de investigación que se pueden seguir se muestran en el capítulo 7.

Al final del documento, se incluyen la bibliografía utilizada y los anexos.

# 2 Simulador OpenHRP3

## 1.4 Descripción general

OpenHRP3 (Open Architecture Humanoid Robotics Plataform version 3) es una plataforma para simulaciones de robots y desarrollo de software. Permite a los usuarios inspeccionar el modelo original del robot y el programa de control a través de una simulación dinámica. Además, OpenHRP3 proporciona diversos componentes software de cálculo y bibliotecas que pueden ser utilizadas para desarrollar software relacionados con la robótica.

OpenHRP3 se desarrolla dentro de un importante programa de investigación impulsado por el Ministerio de Economía e Industria de Japón. Forma parte, como proyecto complementario, del llamado "Distributed component type robot simulator", llevado a cabo por "Cooperation of Next Generation Robots", que pertenece al Gobierno de Cooperación de Ciencia y Tecnología. La ingeniería de cálculos dinámicos es desarrollada por "Nakamura Lab, Dept. of Mechano Informatics, University of Tokyo" y la interfaz gráfica es realizada por "General Robotix, Inc". Las otras partes se desarrollan como trabajo entre cooperación de "Humanoid Resarch Group" y "Task-Intelligence Research Group" en los institutos "Intelligent Systems Research Institute" y "National Institute of Advanced Indrustrial Science and Technology (AIST)" [8].

Esta tercera versión (llamada OpenHRP3) ha mejorado considerablemente el desarrollo en comparación con la anterior versión OpenHRP2. Además, OpenHRP3 se distribuye como software de código abierto.

# 1.5 Requerimientos específicos

## 1.5.1 Programas utilizados

Para utilizar el simulador OpenHRP3 es necesario instalar unos programas específicos que serán descritos a continuación.

### 1.5.1.1 Visual Studio 2008

Microsoft Visual Studio 2008 [10] es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

El nuevo framework (.Net 3.5) está diseñado para aprovechar las ventajas que ofrece el nuevo sistema operativo "Windows Vista" a través de sus subsistemas "Windows Communication Foundation" (WCF) y "Windows Presentation Foundation" (WPF). El primero tiene como objetivo la construcción de aplicaciones orientadas a objetos, mientras que el último apunta a la creación de interfaces de usuario más dinámicas que las conocidas hasta el momento.

A las mejoras de desempeño, escalabilidad y seguridad con respecto a las anteriores versiones, se agregan entre otras, las siguientes novedades:

- La mejora en las capacidades de Pruebas Unitarias permiten ejecutarlas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos. Se incluye además un nuevo soporte para diagnosticar y optimizar el sistema a través de las herramientas de pruebas de Visual Studio. Con ellas se podrán ejecutar perfiles durante las pruebas para que

ejecuten cargas, prueben procedimientos contra un sistema y registren su comportamiento, además de utilizar herramientas integradas para depurar y optimizar.

• Con “Visual Studio Tools for Office” (VSTO) integrado con Visual Studio 2008 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario (UI) de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project.

• Visual Studio 2008 permite incorporar características del nuevo “Windows Presentation Foundation” sin dificultad tanto en los formularios de Windows existentes como en los nuevos. Ahora es posible actualizar el estilo visual de las aplicaciones al de Windows Vista debido a las mejoras en “Microsoft Foundation Class Library (MFC)” y Visual C++. Visual Studio 2008 permite mejorar la interoperabilidad entre código nativo y código manejado por .NET. Esta integración más profunda simplificará el trabajo de diseño y codificación.

• LINQ (Language Integrated Query) es un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos, a través de extensiones para C++ y Visual Basic así como para Microsoft .NET Framework.

• Visual Studio 2008 ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework: 2.0. (Incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con Visual Studio 2008).

• .NET 3.5 incluye biblioteca ASP.NET AJAX para desarrollar aplicaciones web más eficientes, interactivas y altamente personalizadas que funcionen para todos los navegadores más populares y utilicen las últimas tecnologías y herramientas Web, incluyendo Silverlight y Popfly.

En este caso, Visual Studio es el programa encargado de compilar todas las librerías necesarias que están incluidas en el proyecto OpenHRP3.

Algunas de las librerías que posee este proyecto son *Boost*, *Clapack* y *Tvmet*. A continuación se expone una breve explicación de cada una de ellas.

#### 1.5.1.1.1 *Librería BOOST*

BOOST [11] es un conjunto de librerías de código abierto preparadas para extender las capacidades del lenguaje de programación C++. Su licencia permite que sea utilizada en cualquier tipo de proyectos, ya sean comerciales o no.

Su diseño e implementación permiten que sea utilizada en un amplio espectro de aplicaciones y plataformas. Abarca desde librerías de propósito general hasta abstracciones del sistema operativo.

Con el objetivo de alcanzar el mayor rendimiento y flexibilidad se hace un uso intensivo de plantillas. BOOST ha representado una fuente de trabajo e investigación en programación genérica y meta programación en C++.

#### 1.5.1.1.2 *Librería CLAPACK*

En primer lugar se describe *Lapack* debido a que esta librería es la madre de CLAPACK [12]. *Lapack* es una librería de subrutinas escritas en Fortran77 para resolver la mayoría de los problemas que ocurren en álgebra lineal. Ha sido diseñada para ser eficiente en un amplio rango de computadores modernos de alto desempeño. El nombre *Lapack* es el acrónimo de Linear Algebra PACKage.

*Lapack* provee rutinas para resolver sistemas de ecuaciones lineales, problemas de mínimos cuadrados lineales, problemas de valor propio y problemas de valores singulares.

Por otra parte, funcionalidades similares son provistas para matrices reales y complejas, tanto en precisión simple como doble.

La librería CLAPACK fue construida utilizando la herramienta f2c que convierte código escrito en Fortran a código C. Para crear CLAPACK, fue necesario correr la

librería completa de *Lapack* escrita en Fortran77 a través de f2c para obtener código C, y entonces ésta se modificó para mejorar la facilidad de su lectura.
El objetivo principal de CLAPACK consiste en proveer *Lapack* a usuarios que no tienen acceso a compiladores Fortran. Además, gracias a que CLAPACK tiene versiones tanto para LINUX como para Windows, puede ser utilizado por una gran cantidad de usuarios.

#### 1.5.1.1.3 ***Librería TVMET***

TVMET [13] es una librería de vectores y matrices que usan Meta Templates y Expression Templates para evaluar los resultados en el tiempo de compilación.

El código producido es similar al código hand-code, pero la calidad del código todavía depende del compilador y de su versión. Las dimensiones para los vectores y las matrices son estáticas y limitadas.

### 1.5.1.2 Entorno OpenRTM

El OpenRT [9] trabaja la interfaz gráfica interactiva de 3D. Se encarga de los modelos dinámicos animados y de la iluminación global, para la visualización de un prototipo de alta calidad para juegos de ordenador.

El RT-MIDDLEWARE proporciona una plataforma común para la tecnología robótica y aumenta la eficacia de la investigación y desarrollo en la robótica y sus aplicaciones.

Un problema serio a menudo indicado en el desarrollo de software que apoya a los sistemas robóticos comparado con el software tradicional es la imposibilidad de reutilización debido a lo específico de cada sistema robótico. De esta observación, surge la idea de desarrollar un middleware que proporciona una plataforma común para ayudar en el desarrollo de sistemas robóticos, en los cuales un número grande de usuarios podría estar implicado, promoviendo así la reutilización del software de alto nivel en la realización de la tecnología robótica.
El proyecto de investigación y desarrollo de middleware causó varios datos específicos en el nivel de interfaz y la entrega de una puesta en práctica de un prototipo llamado OpenRTM-aist.

Para crear el entorno del proyecto, necesitamos el programa OpenRTM-aist (Advanced Industrial Science and Technology), el cual se ayuda de otros tres programas (ACE, OmniORB y Python). A continuación se hará una descripción de cada uno de ellos.

#### 1.5.1.2.1 *ACE (Adaptative Communication Environment)*

El Ambiente de Comunicación Adaptable (ACE) [14] simplifica varios aspectos de la programación en red. Ofrece un juego de objetos de alto rendimiento orientados a clases de C++ diseñadas para dirigir las complejidades inherentes y desafíos de la programación en red, previniendo errores comunes.

ACE provee un camino estandarizado para el sistema operativo y trabaja con rasgos específicos para ser utilizados. Esto proporciona tipos de datos comunes y métodos para tener acceso a los rasgos poderosos pero complejos de sistemas operativos modernos. Estos incluyen: intercomunicación de los procesos, dirección de hilo, dirección de memoria eficiente, etc.

Fue diseñado para ser portable y proporciona un marco común. El mismo código trabajará sobre la mayor parte de Unix, Microsoft Windows, VxWorks, QNX, OpenVMS etc., con cambios mínimos. Debido a este portabilidad entre plataformas, ACE ha sido usado en el desarrollo de software de comunicación.

El Ambiente de Comunicación Adaptable (ACE) es un marco de código abierto, una herramienta orientada a objetos (OO) que pone en práctica un modelo principal para el software de comunicación simultáneo (concurrente). Las tareas de software de comunicación proporcionadas por ACE incluyen el acontecimiento “demultiplexing” y el envío de eventos; manejan la señal; atienden la inicialización; e interactúan con la comunicación, la dirección de memoria compartida, el envío de mensaje, la configuración dinámica de servicios distribuidos, la ejecución concurrente y la sincronización. ACE está destinado a los servicios de comunicación de alto rendimiento en tiempo real y sus diferentes usos. Esto simplifica el desarrollo de servicios que utilizan la comunicación de interproceso, el acontecimiento demultiplexing, la unión explícita dinámica, y la coincidencia. Además, ACE

automatiza la configuración de sistema y la nueva configuración dinámicamente, uniendo servicios y ejecutando estos servicios en uno o varios procesos o hilos.

#### 1.5.1.2.2 *OmniOrb (Object Request Broker)*

ORB es, en computación distribuida, el nombre que recibe una capa de software (también llamada middleware) que permite a los objetos realizar llamadas a métodos situados en máquinas remotas, a través de una red. Maneja la transferencia de estructuras de datos de manera que sean compatibles entre los dos objetos. Para ello utiliza un estándar para convertir las estructuras de datos en un flujo de bytes, conservando el orden de los bytes entre distintas arquitecturas. Este proceso se denomina marshalling (y también su opuesto, unmarshalling) [15].

Básicamente permite a objetos distribuidos interactuar entre sí de manera transparente, es decir, como si estuviesen en la misma máquina.

OmniORB es un ORB de alto rendimiento y robusto de CORBA para C ++ y Python. Está disponible libremente según los términos de licencia GNU.
En la mayoría de canales de comunicación hay una llamada “inflight” entre dos espacios de dirección en un momento dado. Para hacer esto sin limitar el nivel de coincidencia, se crean nuevos canales que conectan los dos espacios de dirección en demanda cuando hay llamadas concurrentes en curso. Cada canal es servido por un hilo dedicado. Este array proporciona la coincidencia máxima y elimina cualquier hilo que se pone en marcha en cualquiera de los espacios de dirección para tratar una llamada. Además, para maximizar el rendimiento cuando se trata de llamadas grandes, se envían elementos de grandes datos cuando son procesadas mientras otros están siendo ordenados.
De la versión 4.0 en adelante, omniORB también crea una unión de hilo flexible y hace posible varias llamadas en una sola conexión. Esto conlleva una pequeña cantidad de llamadas adicionales, comparado con el hilo por defecto del modelo de conexión, pero permite a omniORB escalar a números sumamente grandes de clientes simultáneos (concurrentes).

#### 1.5.1.2.3 *Python*

Python [16] es un lenguaje de programación dinámico orientado a objetos que puede ser usado para muchas clases de desarrollo de software. Ofrece un gran apoyo a la integración con otros lenguajes e instrumentos y viene con bibliotecas estándar extensas.

Permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK y Qt entre otros.

Los sistemas operativos sobre los que trabaja son Windows, Linux/Unix, Mac OS X, OS/2 y teléfonos móviles Nokia. Python también ha sido puesto a Java y máquinas .NET virtuales. Es distribuido bajo una licencia gratuita para utilizarlo libremente, aún para productos comerciales.

### 1.5.1.3 Java y Jython

#### 1.5.1.3.1 *Java*

Java [17] es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.

Java es una aplicación que nos permite jugar en línea, participar en sesiones de chat con internautas de todo el mundo, calcular los intereses de una hipoteca y ver imágenes en tres dimensiones, entre otras muchas aplicaciones. Es también esencial para las aplicaciones de intranet y otras soluciones de comercio electrónico que constituyen la base informática de las empresas.

Hasta la fecha, la plataforma Java ha atraído a más de cinco millones de desarrolladores de software. Se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, equipos y redes.

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta la han convertido en la tecnología ideal para su aplicación a redes. De portátiles a centros de datos, de consolas de juegos a super equipos científicos, de teléfonos móviles a Internet, Java está en todas partes.

Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Crear programas para que funcionen en un navegador web y en servicios web
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital

En OpenHRP3, algunos de los componentes, tales como GrxUI, están escritos en Java. Por lo tanto, tenemos que configurar el entorno Java para compilar y ejecutar los componentes.

#### 1.5.1.3.2 *Jython*

Jython [18] es una aplicación de alto nivel, orientado a objetos en lenguaje Python escrito en Java. El predecesor de Jython, JPython, está certificado como 100% puro Java. Jython está disponible gratuitamente tanto para aplicaciones comerciales como no comerciales y se distribuye con su código fuente. Jython es complementario a Java y es especialmente adecuado para las siguientes tareas:

- Escritura integrada – Los programadores de Java pueden añadir Jython a bibliotecas de su sistema para permitir a los usuarios finales escribir simples o complicados scripts con el objetivo de añadir la funcionalidad de la aplicación.
- Interactivo con la experimentación - Jython proporciona un intérprete interactivo que puede utilizarse para interactuar con Java, con paquetes, o ejecutar aplicaciones Java. Esto permite a los programadores experimentar y depurar cualquier sistema usando Java Jython.
- El rápido desarrollo de aplicaciones – Los programas Python son típicamente entre 2-10X más corto que el equivalente programa Java. Esto se traduce directamente a un aumento de la productividad del programador. La interacción sin fisuras entre Python y Java permite a los desarrolladores libremente mezclar los dos idiomas.

### 1.5.2 Servidores

OpenHRP3 está formado por distintos servidores encargados de obtener los datos del modelo del robot y del entorno.

### 1.5.2.1 DynamicsSimulator

El simulador dinámico (DynamicsSimulator [8]) calcula la dinámica inversa del link, integrando la aceleración obtenida del joint (articulación) y actualiza la velocidad y valor del joint. Puede manejar arbitrariamente mecanismos en cadena abierta y mecanismos en cadena cerrada. Además, cuando detecta una colisión entre links, y si hay velocidad relativa entre ambos, en primer lugar ajusta la velocidad relativa a cero a través del cálculo de colisión y después calcula la potencia de contacto necesaria para no desarrollar una aceleración relativa. Una parte del resultado del cálculo dinámico puede ser tomado como salida del sensor.

### 1.5.2.2 CollisionDetector

El servidor CollisionDetector [8] detecta el contacto entre el robot y el entorno. Se llama desde el servidor DynamicsSimulator durante la ejecución de la simulación. OPCODE es utilizado como un algoritmo de detección de colisión. El usuario del servidor DynamicsSimulator no necesita llamar al servidor CollisionDetector directamente, sino que es llamado desde el propio DynamicSimulator si es necesario. El CollisionDetector calcula el punto de contacto y el vector de la fuerza de reacción.

### 1.5.2.3 ModelLoader

El ModelLoader [8] lee el modelo de robot con el que trabajará (descrito más adelante) el simulador y extrae los parámetros dinámicos y los datos de dicho robot. El modelo expresa los parámetros dinámicos y la dinámica del robot en el entorno del simulador mediante el uso de archivos VRML.

### 1.5.2.4 VisionSimulator

El servidor VisionSimulator [8] simula el sensor de visión (cámara) que adjunta el robot.

### 1.5.2.5 Controller

Controller [8] es un programa de control que incrementa el lazo de control del robot. Dicho lazo de control garantiza la estabilidad del prototipo durante la realización de tareas y funciona aún cuando el robot está parado.

### 1.5.2.6 CORBA

CORBA [8] (Common Object Request Broker Architecture) es un estándar definido por el Grupo de Dirección de Objeto (OMG) que permite componentes de software escritos en múltiples lenguajes de programación y controla múltiples ordenadores trabajando juntos. Es un mecanismo de software para normalizar la semántica de llamada de método entre los objetos de aplicación que residen en el mismo espacio de dirección (de aplicación) o remoto (el mismo anfitrión, o el anfitrión remoto sobre una red).

OpenHRP3 tiene un sistema de distribución de objetos basado en CORBA. Por este motivo todos los servidores en OpenHRP3 están implementados como objetos CORBA. Cada uno de los servidores están programados en el lenguaje y sistema operativo apropiado y finalmente conectados a través de CORBA.

## 1.5.3 Sistema Operativo

Actualmente OpenHRP3 es soportado en las siguientes plataformas [8].

- Ubuntu Linux 7 o posterior (recomendado).
- Windows XP, Vista (32bit).

## 1.6 Interfaz

En la  se muestra la interfaz del simulador OpenHRP3.

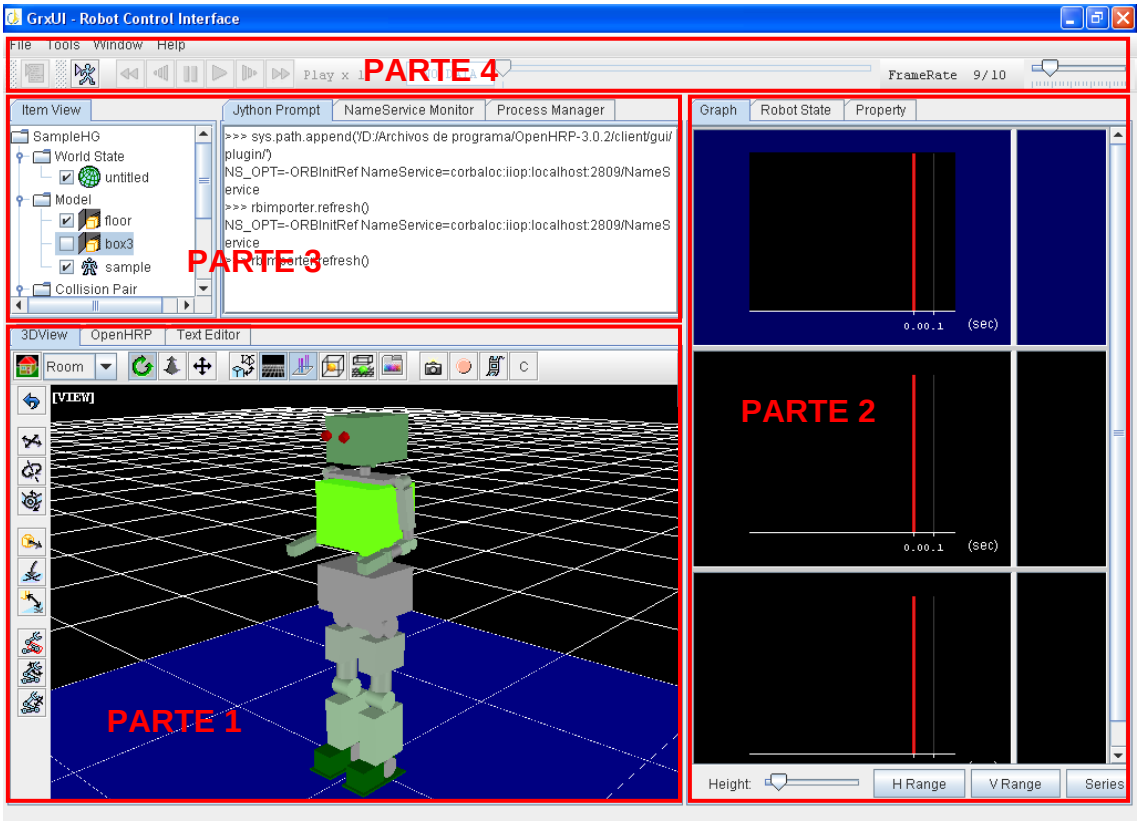


**Figura 2.1 Interfaz simulador OpenHRP3**

Podemos dividir la interfaz gráfica en cuatro partes. La primera de ellas está dirigida a la visualización del modelo del robot durante la simulación (Figura 2.2).

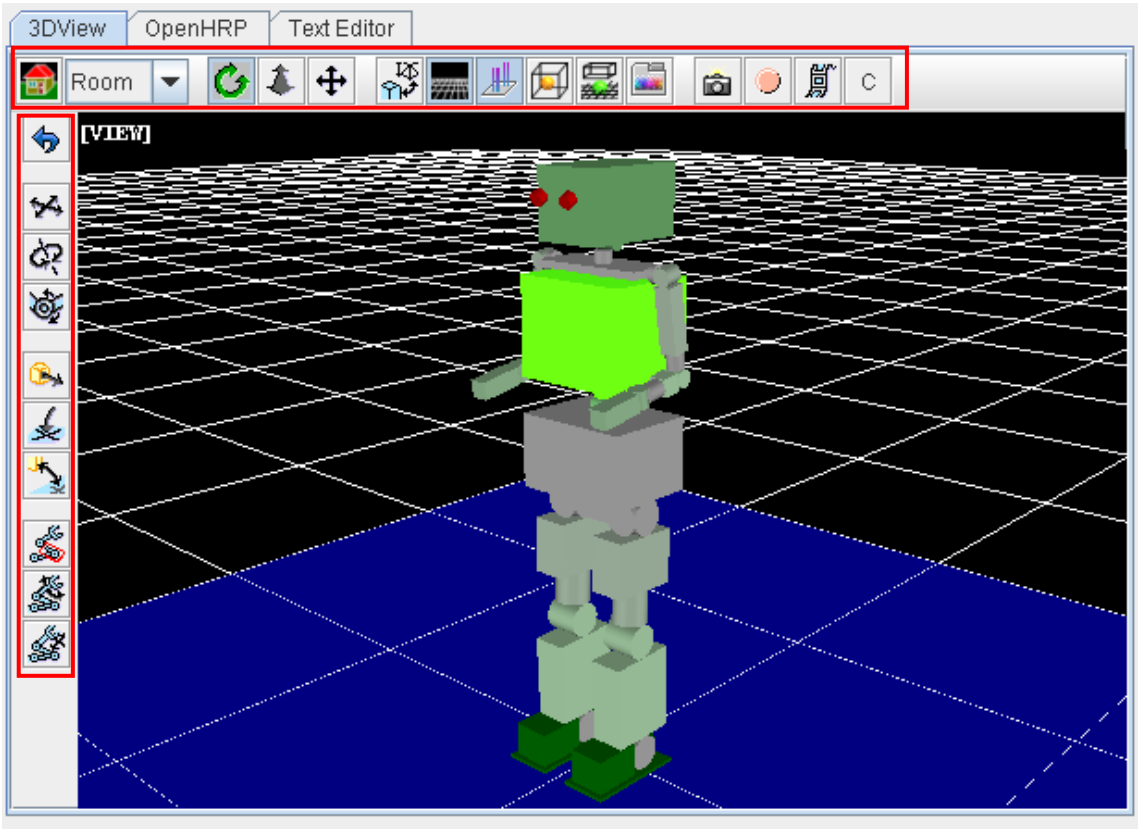


**Figura 2.2 Visualización modelo**

En esta zona se encuentran dos barras de herramientas, una vertical y otra horizontal. Las herramientas dispuestas horizontalmente nos permiten cambiar la vista del robot durante la simulación mediante los 3 ejes de coordenadas, ocultar planos de las figuras, ver los centros de gravedad del robot o de los objetos introducidos, además de tomar fotografías y videos de la simulación. Las verticales sin embargo, están orientadas al control de movimiento de una simple articulación, rotación de objetos determinados, comprobación de cinemáticas inversas de las figuras, etc.

La parte número 2, situada a la derecha de la interfaz, se visualizan las gráficas (Figura 2.3) que representan la evolución de las variables articulares durante la simulación (posición, velocidad, aceleración), así como los datos (Figura 2.3) de los distintos sensores y articulaciones incluidos a lo largo de la estructura mecánica del robot (sensores de fuerza, etc.).

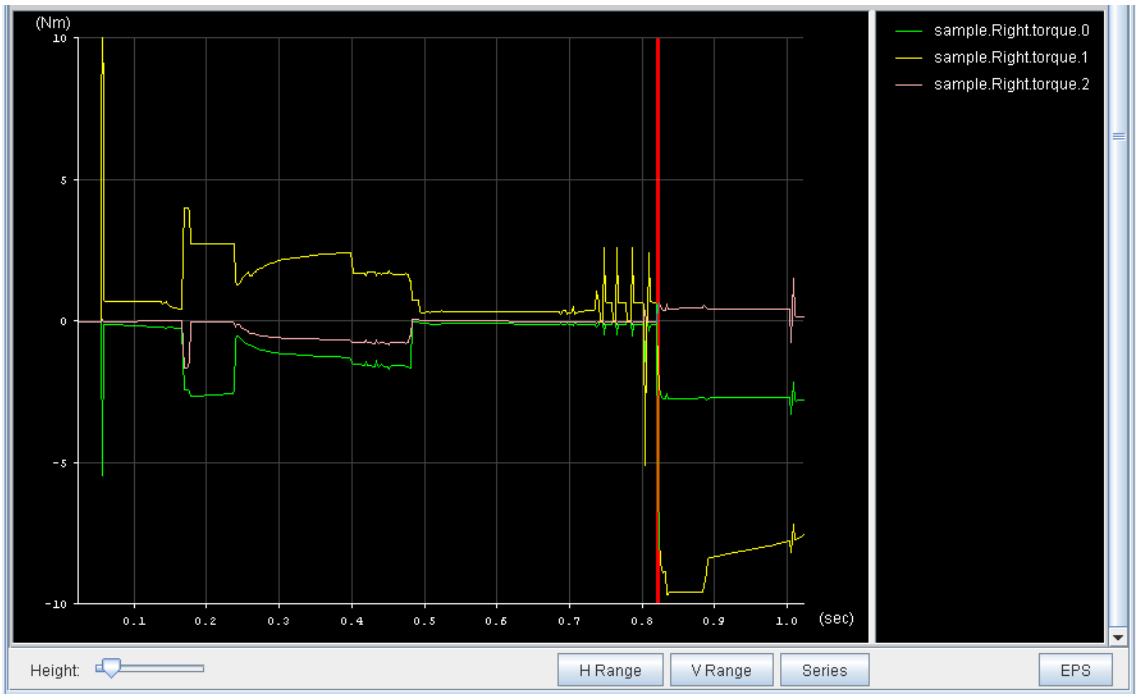


**Figura 2.3 Vista de las gráficas**

Graph | Robot State | Property

| No | Joint | Angle | Target | Current | PWR | SRV | Pgain | Dg |
|---|---|---|---|---|---|---|---|---|
| 0 | RLEG_HIP_R | 0.0 | --- | --- | --- | --- | --- | -- |
| 1 | RLEG_HIP_P | -2.1 | --- | --- | --- | --- | --- | -- |
| 2 | RLEG_HIP_Y | 0.0 | --- | --- | --- | --- | --- | -- |
| 3 | RLEG_KNEE | 4.5 | --- | --- | --- | --- | --- | -- |
| 4 | RLEG_ANKLE_P | -2.4 | --- | --- | --- | --- | --- | -- |
| 5 | RLEG_ANKLE_R | 0.0 | --- | --- | --- | --- | --- | -- |
| 6 | RARM_SHOULD... | 10.0 | --- | --- | --- | --- | --- | -- |
| 7 | RARM_SHOULD... | -0.2 | --- | --- | --- | --- | --- | -- |
| 8 | RARM_SHOULD... | 0.0 | --- | --- | --- | --- | --- | -- |
| 9 | RARM_ELBOW | -90.0 | --- | --- | --- | --- | --- | -- |
| 10 | RARM_WRIST_Y | 0.0 | --- | --- | --- | --- | --- | -- |
| 11 | RARM_WRIST_R | 0.0 | --- | --- | --- | --- | --- | -- |
| 12 | LLEG_HIP_R | 0.0 | --- | --- | --- | --- | --- | -- |
| 13 | LLEG_HIP_P | -2.1 | --- | --- | --- | --- | --- | -- |
| 14 | LLEG_HIP_Y | 0.0 | --- | --- | --- | --- | --- | -- |
| 15 | LLEG_KNEE | 4.5 | --- | --- | --- | --- | --- | -- |
| 16 | LLEG_ANKLE_P | -2.4 | --- | --- | --- | --- | --- | -- |
| 17 | LLEG_ANKLE_R | 0.0 | --- | --- | --- | --- | --- | -- |
| 18 | LARM_SHOULD... | 10.0 | --- | --- | --- | --- | --- | -- |
| 19 | LARM_SHOULD... | -0.2 | --- | --- | --- | --- | --- | -- |
| 20 | LARM_SHOULD... | 0.0 | --- | --- | --- | --- | --- | -- |
| 21 | LARM_ELBOW | -90.0 | --- | --- | --- | --- | --- | -- |
| 22 | LARM_WRIST_Y | 0.0 | --- | --- | --- | --- | --- | -- |
| 23 | LARM_WRIST_R | 0.0 | --- | --- | --- | --- | --- | -- |
| 24 | WAIST_P | 0.0 | --- | --- | --- | --- | --- | -- |
| 25 | WAIST_Y | 0.0 | --- | --- | --- | --- | --- | -- |
| 26 | CHEST_Y | 0.0 | --- | --- | --- | --- | --- | -- |
| 27 | CHEST_P | 0.0 | --- | --- | --- | --- | --- | -- |

**Figura 2.4 Vista de los datos de articulaciones**

En la parte número 3 de la interfaz se puede observar el árbol de módulos cargados en la simulación, tal y como se muestra en la parte izquierda de la Figura 2.5.

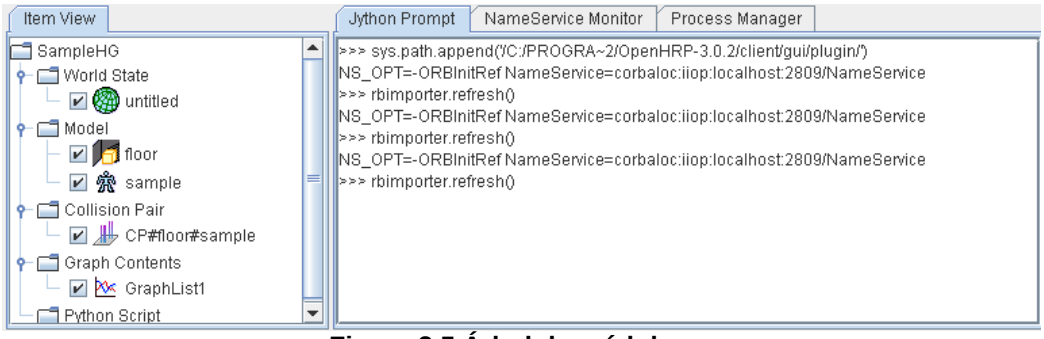


**Figura 2.5 Árbol de módulos**

Estos módulos pueden ser el modelo del robot, el modelo del entorno, el módulo de colisión (Collision Pair) y el paquete de visualización de gráficas, entre otros. En la derecha del árbol de módulos se encuentra la ventana “Jython Prompt”, donde se pueden introducir líneas de comando; la ventana “Process Manager”, donde podemos ver todos los procesos cargados por la interfaz de simulación; o la ventana “NameService Monitor” donde se observan los registros del servidor CORBA.

Por último se tiene la parte número 4, donde se puede manejar el inicio de la simulación, la parada de la misma, una vez obtenida se puede ver el resultado de la misma a una velocidad mayor; podemos cambiar el “frame rate”, el número de frames que se quiere que contenga el video de la simulación (en caso de querer grabarlo); además del menú donde podemos guardar el proyecto actual, crear uno nuevo, guardarlo, etc.


**Figura 2.6 Barra de menú y controles de simulación**

## 1.7 Arquitectura del simulador

Para entender mejor el funcionamiento del simulador y la relación que existe entre los diferentes archivos, etc, se adjunta un diagrama de bloques en el que se explica cual es el proceso de ejecución de un proyecto determinado, cabe destacar que muchas de las funcionalidades mencionadas en el diagrama serán explicadas en el Capítulo 4: Simulación de tareas en OpenHRP3 [4]:

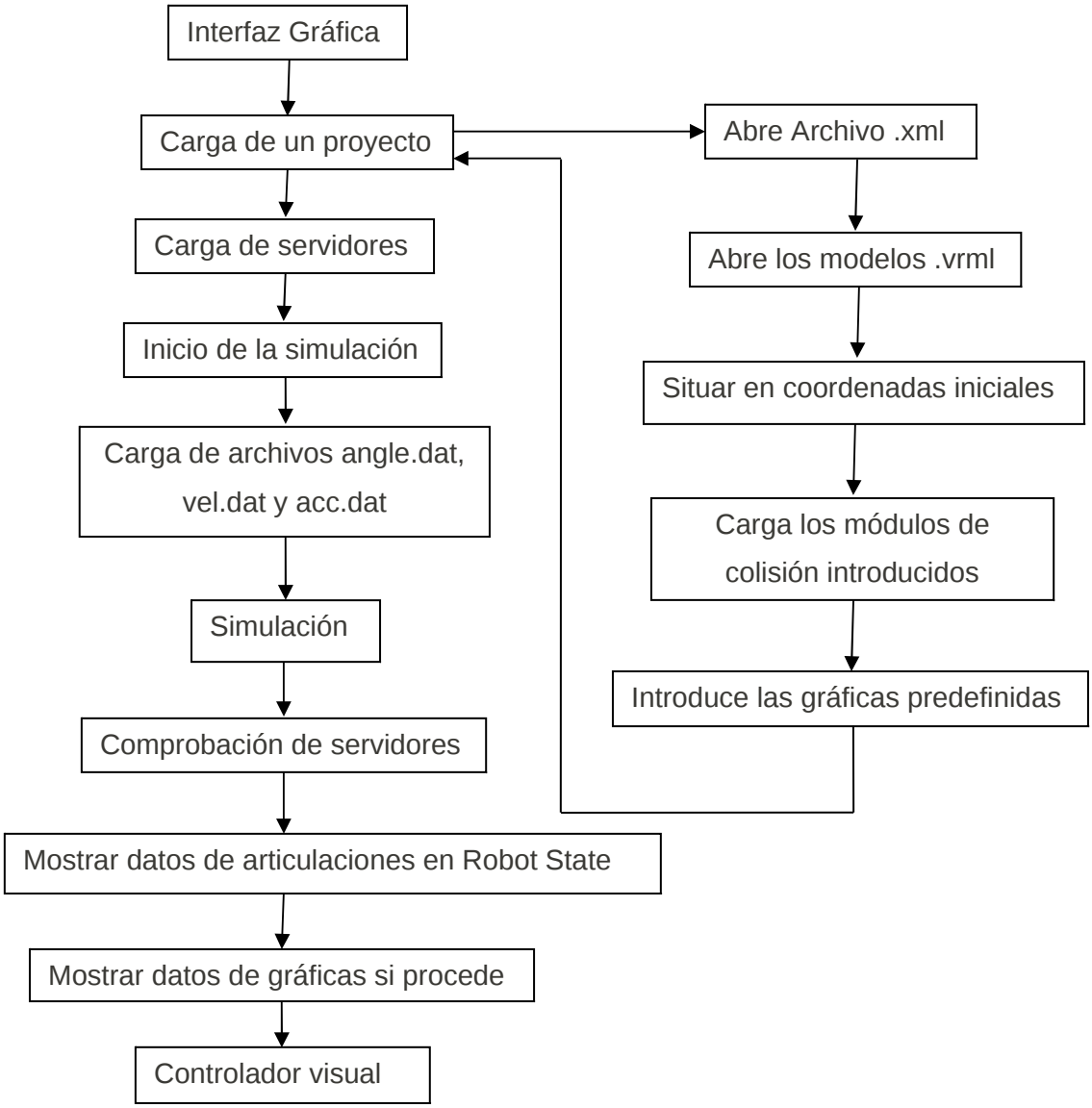


**Figura 2.7 Arquitectura del Simulador**

# 2 Modelo de la plataforma RH-2 en OpenHRP3

## 2.1 RH-2

Las medidas del Rh-2 son las que se indican en la figura 3.1.

Estas medidas están adaptadas a nuestro proyecto, para asemejar lo más posible la simulación a la realidad. Dado que el simulador no acepta formas ovaladas, ni elípticas, el robot ha sido creado a semejanza con dos tipos de formas: cilíndricas y prismáticas. Se han respetado tanto las medidas de las piernas, como la de los brazos, y altura y anchura del robot, puesto que el tronco del robot consta de cubiertas más anchas.

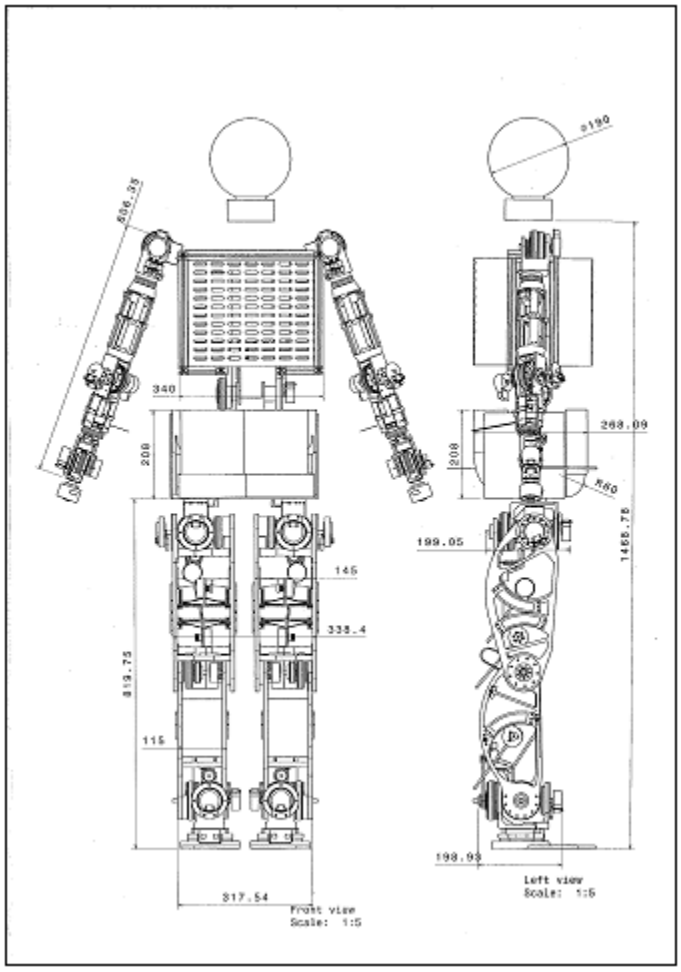


**Figura 3.8 Dimensiones del robot**

## 2.2 Modelo VRML del robot RH-2

El lenguaje VRML es pieza fundamental de la representación virtual que realiza la plataforma de simulación OpenHRP3. Para el modelo desarrollado del robot RH-2 se ha utilizado la versión 2.0 de dicho lenguaje. El modelo se muestra en la Figura 3.2.

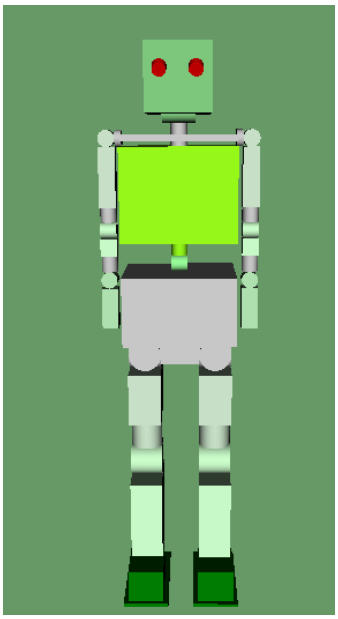

**Figura 3.9 Modelo VRML robot RH-2**

### 2.2.1 Estructura del fichero VRML

El fichero VRML desarrollado para el modelo en el simulador OpenHRP3 [6] consta de unos elementos básicos:

- Cabecera
- Comentarios
- Nodos

La cabecera indica el estándar empleado, la versión del archivo VRML y el uso de caracteres internacionales. En este caso la cabecera del archivo es: #VRML V2.0 utf8. Es importante resaltar que no debe existir ningún espacio en blanco entre el símbolo "#" y la palabra "VRML".

Los comentarios en VRML se escriben en una sola línea, comenzándola con el símbolo "#". Se pueden incluir tantas líneas de comentarios como se desee.

Un nodo es la estructura mínima indivisible de un fichero VRML y tiene como misión la definición de las características de un objeto o bien las relaciones entre distintos objetos. Los nodos contienen campos que describen propiedades de los objetos. Todo campo tiene un tipo determinado y no se puede inicializar con valores de otro tipo. De este modo, cada tipo de nodo tiene una serie de valores predeterminados para todos

sus campos, de forma que cuando se utilice solo debe indicarse aquellos campos que se quieran modificar.

Existen 5 tipos de nodo, habiendo uno principal que engloba los demás:

- El nodo PROTO, que consta de Humanoid, Joint, Segment, y sensores.
    - El nodo Joint, que define cada articulación del robot, al que irán asociados los eslabones correspondientes definidos en el nodo Segment.
    - El nodo Segment. Éste define la forma de la pieza a la que está asociada, pudiendo ser un cilindro o un prisma.
    - El nodo Humanoid, el cuál es el nodo raíz del modelo.
    - Nodos de sensores. VisionSensor, ForceSensor, Gyro, AccelerationSensor y PressureSensor. Se utilizan para medir visión, fuerza, giro, aceleración y presión, respectivamente. Pueden llevar asociado un Segment para darle forma (por ejemplo los ojos) o simplemente situados en los puntos clave.

A continuación se muestra en detalle el robot, con todos sus joints para cada grado de libertad.

Después de cada articulación, hay una letra: R, P o Y.

- R significa ROLL, y viene dado por que la articulación gira en el eje X.
- P viene de PITCH, y es porque la articulación gira respecto al eje Y.
- Y es YAW, lo que quiere decir q el eje de giro de la articulación es el Z.

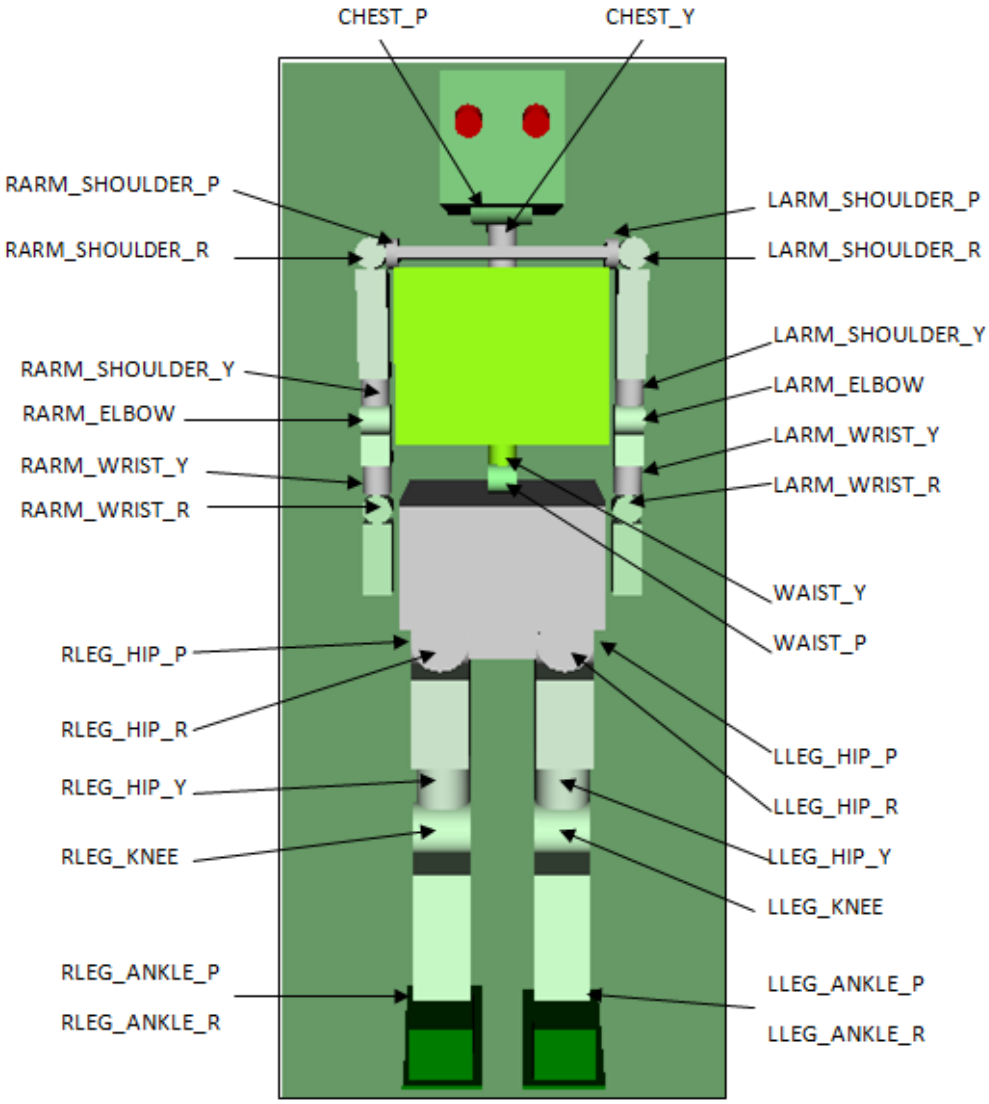


**Figura 3.10 Articulaciones del RH-2 en VRML**

# 3 Simulación de tareas en OpenHRP3

## 3.1 Archivos necesarios para la simulación

Para realizar simulaciones en la plataforma OpenHRP3 son necesarios unos determinados archivos, los cuales definen el modelo del robot completamente.

Uno de estos archivos es el modelo VRML del robot, nombrado anteriormente. Por otra parte, también son necesarios los archivos donde se encuentra definida la trayectoria del robot que se desea simular, el archivo del controlador y el archivo del proyecto. Cada uno de estos archivos se definirá a continuación.

### 3.1.1 Archivos de trayectorias

La trayectoria del paso del robot viene indicada por tres archivos, ubicados en la carpeta: C:\Program Files\OpenHRP-3.0.2\Controller\rtc\SampleHG\etc. Estos archivos son *vel.dat*, *acc.dat* y *angle.dat,* e indican las velocidades, aceleraciones y ángulos de cada una de las articulaciones (joints).

Las unidades de los datos del archivo donde se indican los ángulos son radianes. Así, en el archivo donde se definen las velocidades las unidades son rad/seg y por último en el archivo donde se indican las aceleraciones las unidades serán $rad/seg^2$ .

Los datos vienen indicados en columnas, siendo la primera la de la base de tiempos (en este caso 0.005), la segunda equivale al joint número cero, la tercera al número uno, y así sucesivamente. Por esta razón los archivos constan de 29 columnas, debido a los 28 joints o grados de libertad (del 0 al 27) del Rh-2.

En la Figura 4.1 se muestra las cuatro primeras columnas de los ángulos del paso pertenecientes a los tres ejes de la cadera derecha y la rodilla derecha (las columnas con valores son la rodilla y la cadera en el eje “y” cuando el robot está flexionando).

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0.005 | 0 | -7.13E-06 | 0 | 1.43E-05 |
| 0.01 | 0 | -2.84E-05 | 0 | 5.69E-05 |
| 0.015 | 0 | -6.39E-05 | 0 | 0.00012774 |
| 0.02 | 0 | -0.00011332 | 0 | 0.00022664 |
| 0.025 | 0 | -0.00017671 | 0 | 0.00035342 |
| 0.03 | 0 | -0.00025395 | 0 | 0.0005079 |
| 0.035 | 0 | -0.00034495 | 0 | 0.00068991 |
| 0.04 | 0 | -0.00044964 | 0 | 0.00089928 |
| 0.045 | 0 | -0.00056793 | 0 | 0.00113585 |
| 0.05 | 0 | -0.00069972 | 0 | 0.00139945 |
| 0.055 | 0 | -0.00084495 | 0 | 0.0016899 |
| 0.06 | 0 | -0.00100352 | 0 | 0.00200704 |
| 0.065 | 0 | -0.00117535 | 0 | 0.0023507 |
| 0.07 | 0 | -0.00136035 | 0 | 0.0027207 |
| 0.075 | 0 | -0.00155845 | 0 | 0.00311689 |
| 0.08 | 0 | -0.00176955 | 0 | 0.0035391 |
| 0.085 | 0 | -0.00199357 | 0 | 0.00398715 |
| 0.09 | 0 | -0.00223044 | 0 | 0.00446088 |
| 0.095 | 0 | -0.00248006 | 0 | 0.00496013 |

**Figura 4.11 Estructura del archivo con extensión dat.**

### 3.1.2 Archivo del controlador

Además de los tres archivos donde se define la trayectoria, también debemos configurar adecuadamente el controlador (este archivo se encuentra en el directorio: C:\Program Files\OpenHRP 3.0.2\Controller\rtc\SampleHG\SampleHG.cpp).

Una de las funciones del controlador es ir leyendo los valores de los archivos donde se encuentran los ángulos, velocidades y aceleraciones. Esto lo hace mediante un bucle *for*, el cual lee los valores de los archivos basándose en la variable *DOF*, a la cual hay que asignarle un valor para que recorra los archivos desde la primera columna hasta la última, eso sí, sin contar la base de tiempos. Por esta razón, en nuestro caso es 28 (de 0 a 27 grados de libertad).

```
#define DOF (28)

 ............
 ............
 ............

for (i=0; i<DOF; i++)
  {
      angle >> m_angle.data[i];
      vel   >> m_vel.data[i];
      acc   >> m_acc.data[i];
```

**Figura 4.12 Muestra del código del archivo del controlador.**

### 3.1.3 Archivos de proyecto

Otro de los archivos necesarios para la simulación es el archivo que configura el proyecto que vamos a simular. Este archivo tiene extensión .XML.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<grxui>
    <mode name="Simulation">
        <item class="com.generalrobotix.ui.item.GrxWorldStateItem" name="untitled" select="true">
            <property name="logTimeStep" value="0.0050 "/>
            <property name="integrate" value="true"/>
            <property name="viewsimulate" value="false"/>
            <property name="totalTime" value="7.0 "/>
            <property name="timeStep" value="0.0050 "/>
            <property name="method" value="RUNGE_KUTTA"/>
            <property name="gravity" value="9.8 "/>
            <property name="viewsimulationTimeStep" value="0.033 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem" name="floor" select="true" url="$(OPENHRPHOME)/etc/floor.wrl">
            <property name="isRobot" value="false"/>
            <property name="WAIST.rotation" value="0.0 1.0 0.0 0.0 "/>
            <property name="WAIST.translation" value="0.0 0.0 -0.1 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem" name="box3" select="true" url="$(OPENHRPHOME)/etc/box3.wrl">
            <property name="isRobot" value="false"/>
            <property name="WAIST.rotation" value="0.0 0.0 0.0 0.0 "/>
            <property name="WAIST.translation" value="0.32 0.0 1.65 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem" name="sample" select="true" url="$(OPENHRPHOME)/etc/sample.wrl">
            <property name="RLEG_HIP_R.angle" value="0.0 "/>
            <property name="RARM_SHOULDER_R.mode" value="HighGain"/>
            <property name="LLEG_KNEE.mode" value="HighGain"/>
            <property name="RARM_ELBOW.angle" value="-1.5708 "/>
            <property name="LLEG_ANKLE_P.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_P.angle" value="-0.0424675 "/>
            <property name="LLEG_ANKLE_R.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_R.angle" value="0.0 "/>
            <property name="LLEG_HIP_Y.mode" value="HighGain"/>
            <property name="RLEG_HIP_P.mode" value="HighGain"/>
            <property name="RARM_WRIST_P.angle" value="0.0 "/>
            <property name="CHEST.mode" value="HighGain"/>
            <property name="RARM_WRIST_R.angle" value="0.0 "/>
            <property name="RARM_WRIST_Y.angle" value="0.0 "/>
            <property name="LLEG_KNEE.angle" value="0.0785047 "/>
            <property name="LLEG_HIP_R.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_Y.angle" value="0.0 "/>
            <property name="LARM_WRIST_P.angle" value="0.0 "/>
            <property name="LARM_WRIST_R.angle" value="0.0 "/>
            <property name="LARM_WRIST_Y.angle" value="0.0 "/>
            <property name="RARM_WRIST_Y.mode" value="HighGain"/>
            <property name="RARM_ELBOW.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_Y.mode" value="HighGain"/>
```

**Figura 4.13 Ejemplo de archivo XML**

Como se muestra en la Figura 4.3, en este archivo se especifican los parámetros de simulación como la base de tiempo, el tiempo de duración de la simulación, etc. Además, también se especifica qué objetos VRML van a ser cargados en el proyecto, por ejemplo el modelo del robot, suelo, cajas, etc. En todos ellos se puede indicar la

posición inicial del objeto en el visor del simulador. Además, en el modelo del robot se pueden definir valores iniciales de cada una de las articulaciones que componen el modelo.

También se puede definir la interacción entre dos objetos cargados en el proyecto, así como las constantes de rozamiento estático y dinámico entre dichos objetos.

## 3.2 Cómo cargar el proyecto en el simulador

En primer lugar se debe ir al directorio que contiene la interfaz del simulador, ver Figura 4.4. Este directorio es **OpenHRP-3.0.2\bin\dos**. A continuación se hace doble clic sobre el archivo **GrxUI.bat**.

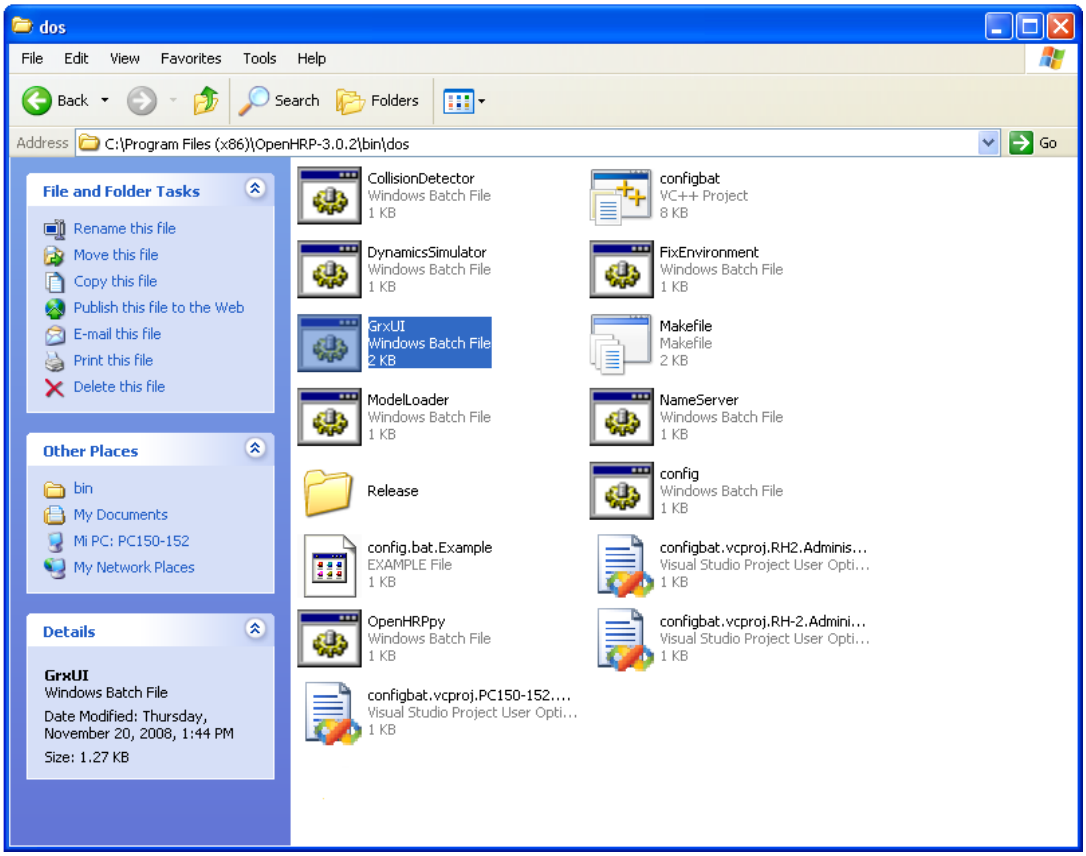


**Figura 4.14 Directorio de la interfaz del simulador.**

Cuando la interfaz del simulador esté abierta se debe cargar el proyecto que se desea simular. Para ello seleccionar el menú File y después Load Project como se muestra en la Figura 4.5.

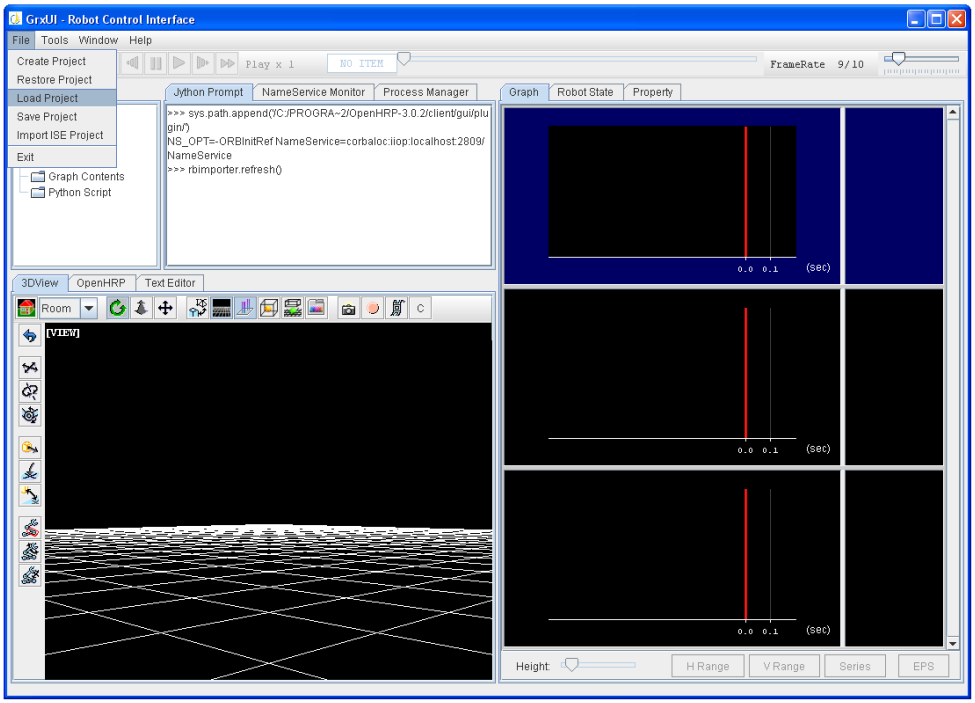


**Figura 4.15 Cargar proyecto en el simulador.**

A continuación, aparece una ventana donde elegimos el proyecto que se desea cargar (ver Figura 4.6).

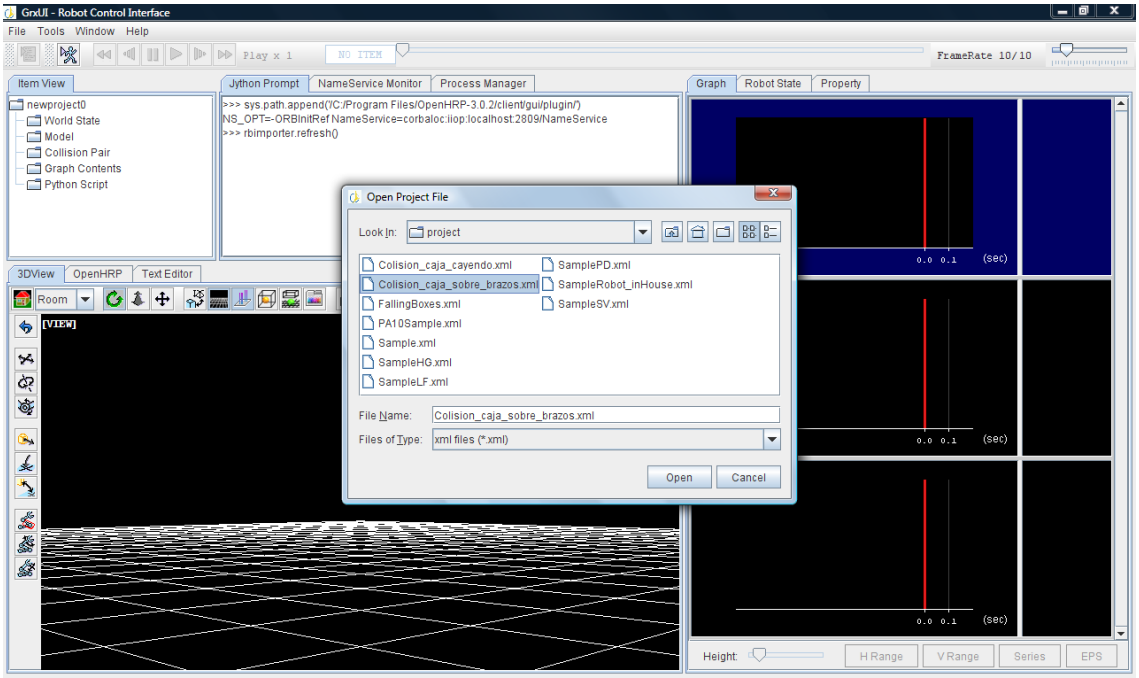


**Figura 4.16 Seleccionar proyecto.**

Una vez cargado el proyecto correctamente nos deben aparecer los objetos componen dicho proyecto en el simulador, en la parte correspondiente al árbol de módulos (ver Figura 4.7).

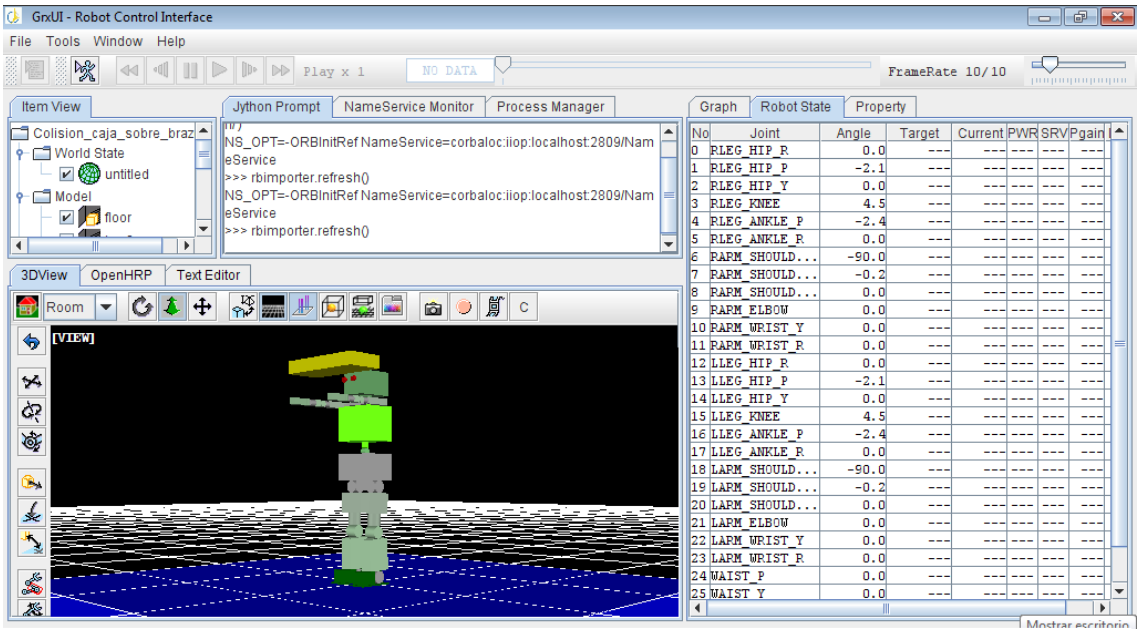


**Figura 4.17 Proyecto cargado.**

Para comenzar la simulación se debe pulsar el botón "Start Simulation", que viene resaltado en la Figura 4.8.

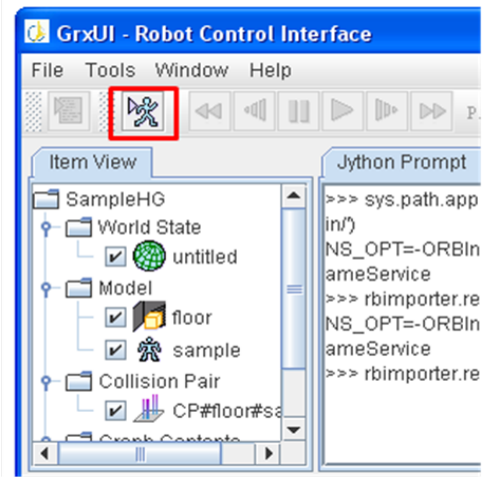


**Figura 4.18 Iniciar simulación.**

Para suspender o finalizar la simulación se debe pulsar el botón "Suspend Simulation". Este botón es el mismo que el de “Start Simulation” pero al iniciar ha cambiado el icono.

Para observar los datos de los sensores en las gráficas en tiempo real, se debe seleccionar el botón "Series" que se encuentra en la parte inferior derecha de la interfaz, como muestra la Figura 4.9 [2].

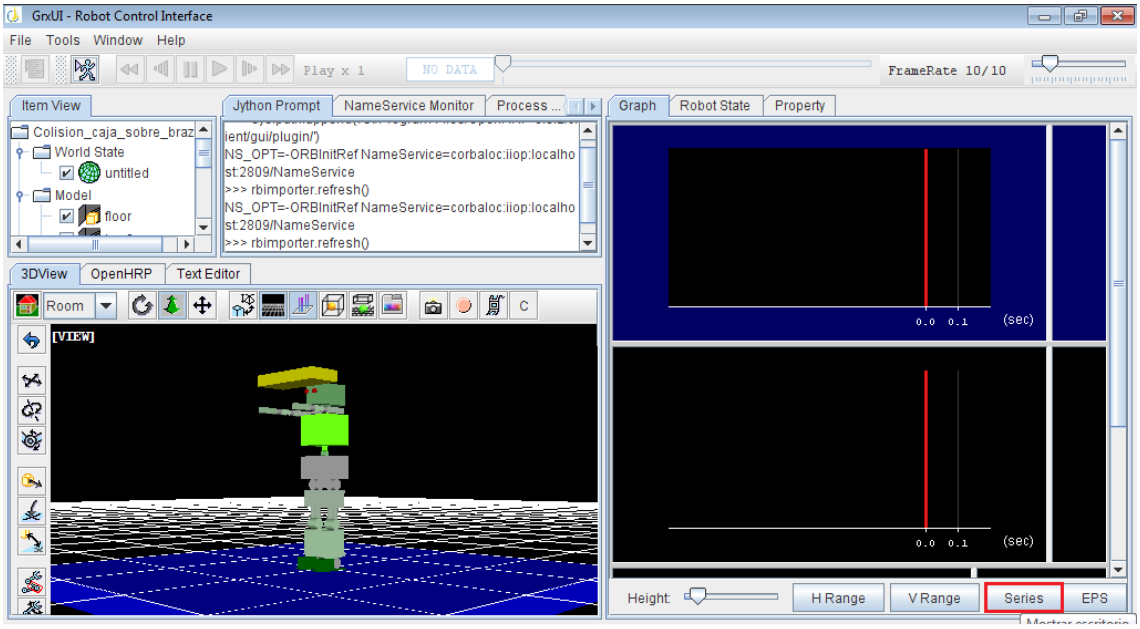


**Figura 4.19 Insertar gráfica.**

Después de pulsar el botón "Series" aparece un cuadro de diálogo donde se elige el tipo de sensor (Figura 4.10), el nombre del sensor (Figura 4.11) que se incluye en el modelo que se quiere observar y el atributo (Figura 4.12). Nos aparecerá el sensor deseado arriba (Figura 4.13). Una vez realizado esto se debe pulsar el botón "Set" y a continuación el botón "Ok".

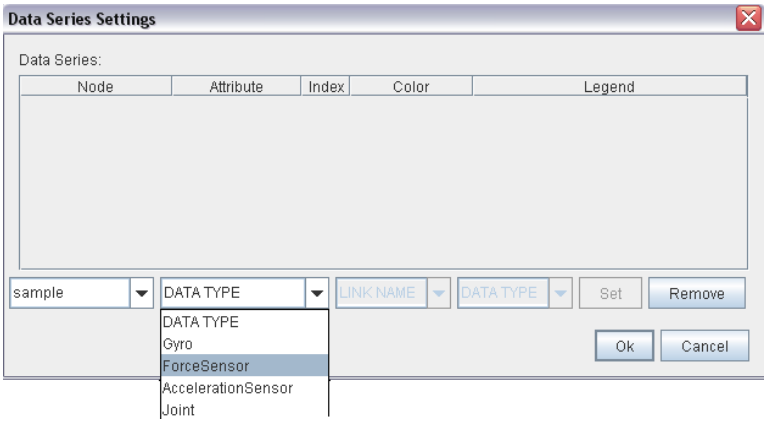


**Figura 4.20 Selección del tipo de sensor**

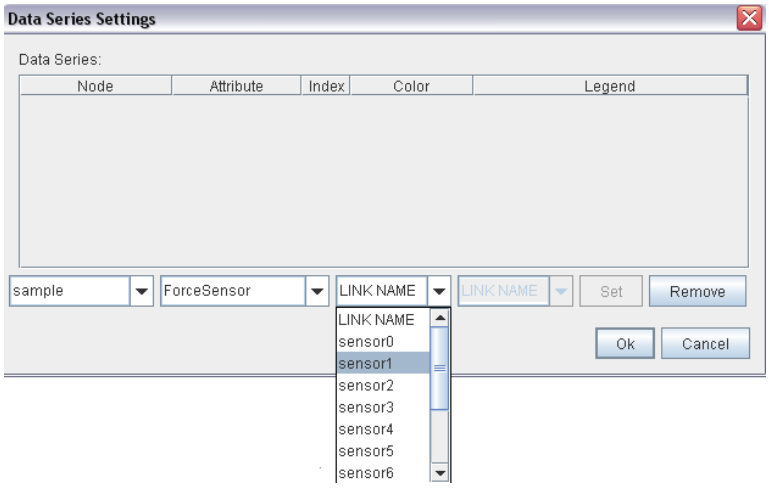


**Figura 4.21 Selección del nombre del sensor**

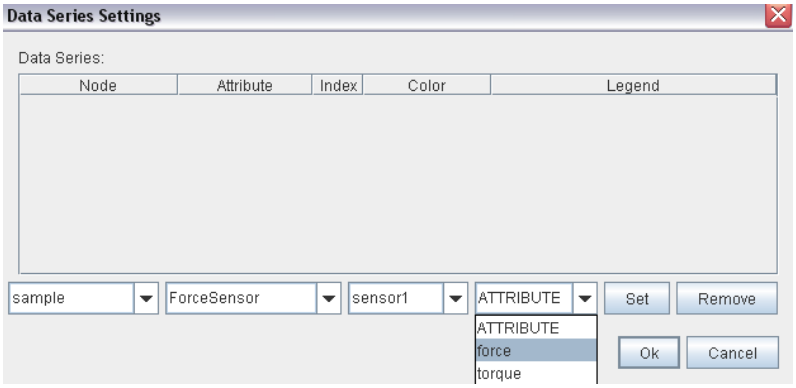


**Figura 4.22 Selección de atributo**

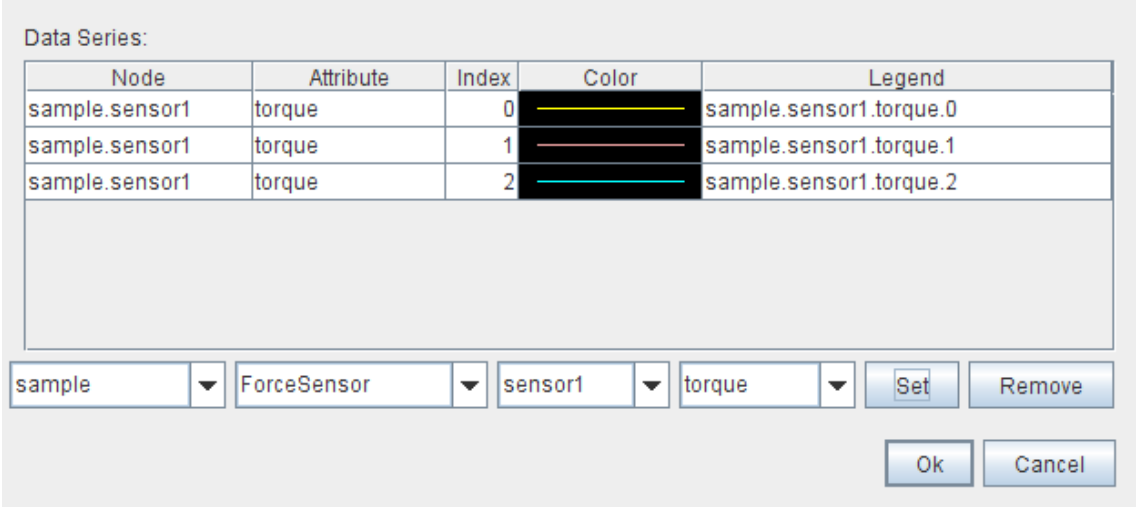


**Figura 4.23 Configuración gráfica.**

Otra posibilidad que nos ofrece OpenHRP3 es guardar en un archivo con extensión “csv” todos los datos obtenidos de la simulación, como las traslaciones y rotaciones de las articulaciones y también todos los datos que se obtienen de los sensores que se incluyen en el modelo. Para ello, una vez realizada la simulación se selecciona el elemento del árbol que hace referencia a World State que en este caso recibe el nombre de untitled (ver Figura 4.14).

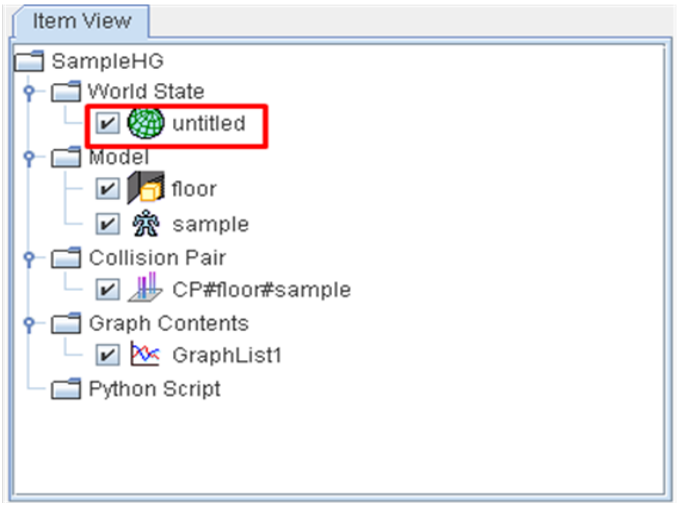


**Figura 4.24 Elemento untitled.**

Una vez seleccionado, se hace click sobre el elemento con el botón derecho del ratón y se selecciona "saveAsCSV", como se muestra en la Figura 4.15.

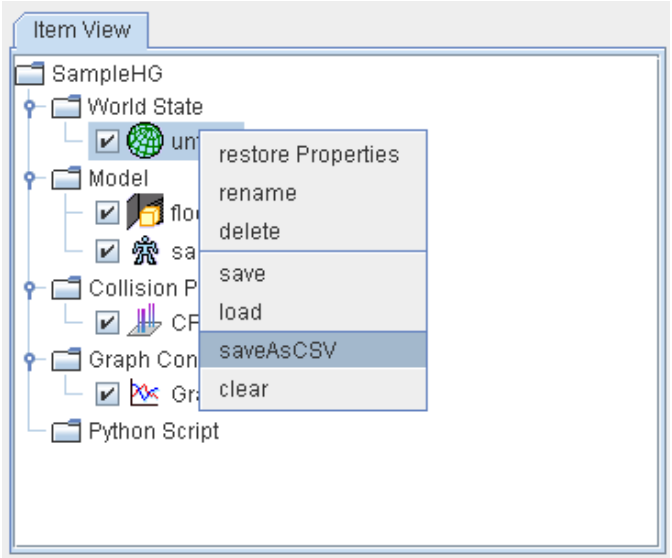

**Figura 4.25 Selección saveAsCSV.**

Como se muestra en la Figura 4.16 este archivo .csv se guarda en el siguiente directorio: C:\Program Files\OpenHRP-3.0.2\client\gui\log.

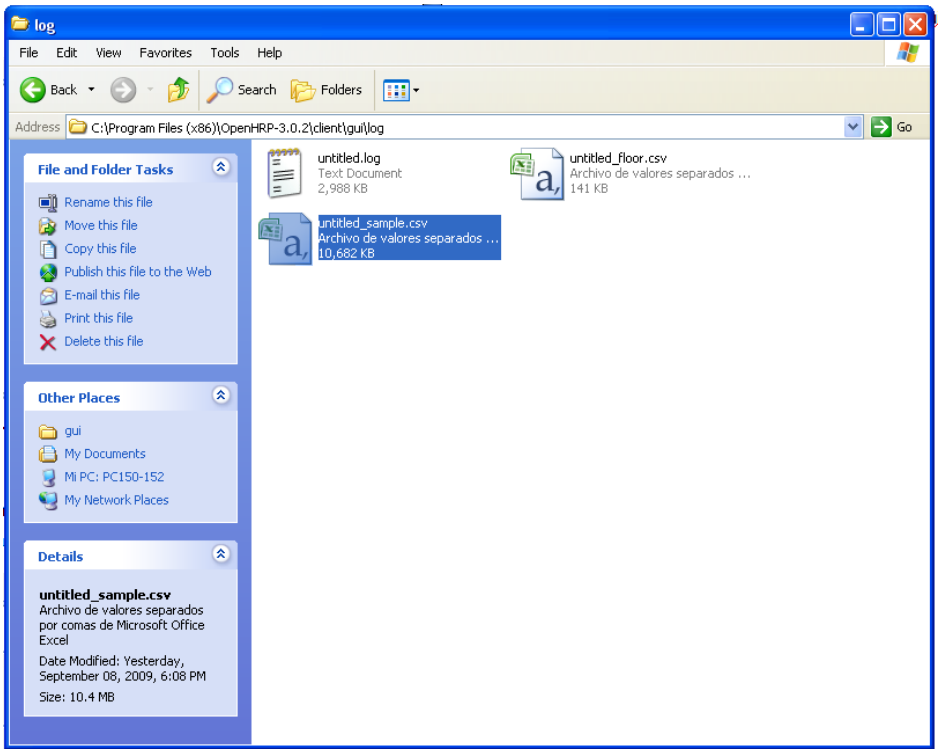


**Figura 4.26 Ubicación archivo .csv.**

# 4 Simulación de tareas del Rh-2 en OpenHRP3

## 4.1 INTRODUCCIÓN

Lo que se pretende demostrar con estas simulaciones es la variación de pares de fuerza que sienten los sensores situados en los brazos y los tobillos al depositar una caja sobre ellos, modificando la masa de la misma. También es importante ver cómo y cuánto varía la normalidad del paso que, sin caja, es estable.

Es importante señalar que las imágenes mostradas denotan un error tras realizarse una fusión de la caja con el brazo, y las líneas que se disparan no influyen en los datos, puesto que únicamente representan la interacción entre los modelos (ver figura 5.1).

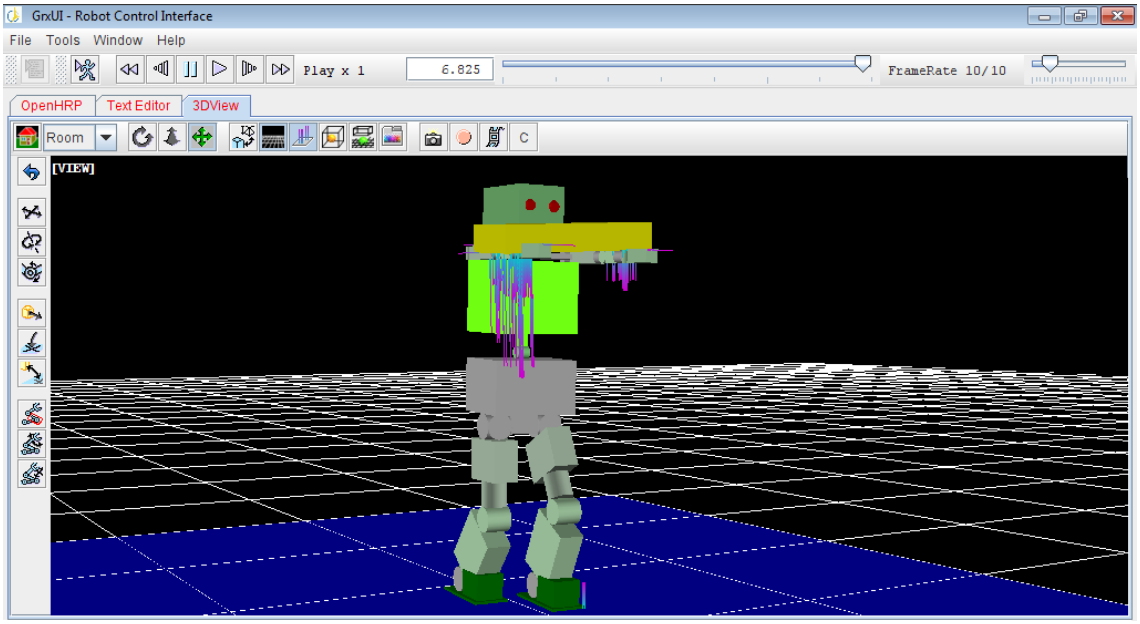


**Figura 5.27 Caja fusionándose con brazo derecho**

También se comprobará si los datos son coherentes, probando una caja de distinto peso y viendo que los pares son proporcionales. Esto se hará mediante una caja de 3 Kg y otra de 30 Kg, viendo que el aumento de los pares es diez veces mayor.

Debe puntualizarse que los pares representados en las gráficas están en Newton por metro (N*m).
Los sensores están situados (no físicamente) en el centro de la articulación a la que pertenecen.
Cada sensor consta de 3 ejes en los que siente las fuerzas, torque[0], torque[1] y torque[2], los cuales están indicados en la figura 5.2.

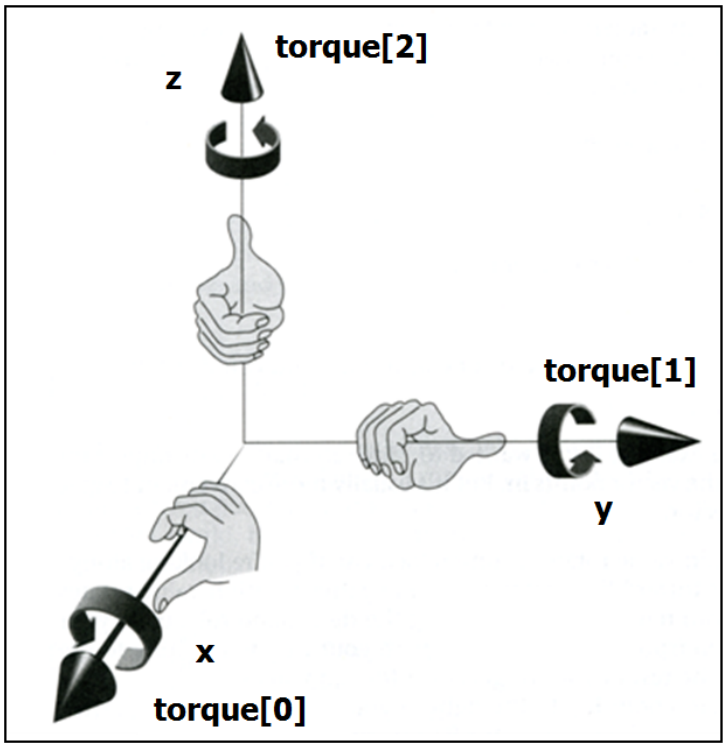


**Figura 5.28 Ejes de los pares de fuerza**

Se van a estudiar tres casos:

- En el primer caso se estudiará el paso del robot sin caja, con los brazos extendidos. En este ejemplo, el robot termina el paso de manera estable. Esto se realiza para calibrar al robot sin fuerzas externas.
  Se debe hacer reseña en el peso de cada brazo del robot, el cual es de 6 Kg.
- En el segundo, el robot andará sosteniendo una caja de 3 Kg. Tras realizar la misma simulación que en el primer caso, se observa una similitud en la representación del simulador, puesto que el robot soporta sin ningún problema las fuerzas que la caja ejerce sobre él. Gráficamente, se aprecia la variación en los pares. Ver figura 5.3.
- El tercer caso se simula con una caja de 30 Kg, viéndose cómo el robot, al finalizar el segundo paso, se desestabiliza y cae, además de caérsele la caja de entre los brazos debido al elevado peso. Ver figura 5.4.

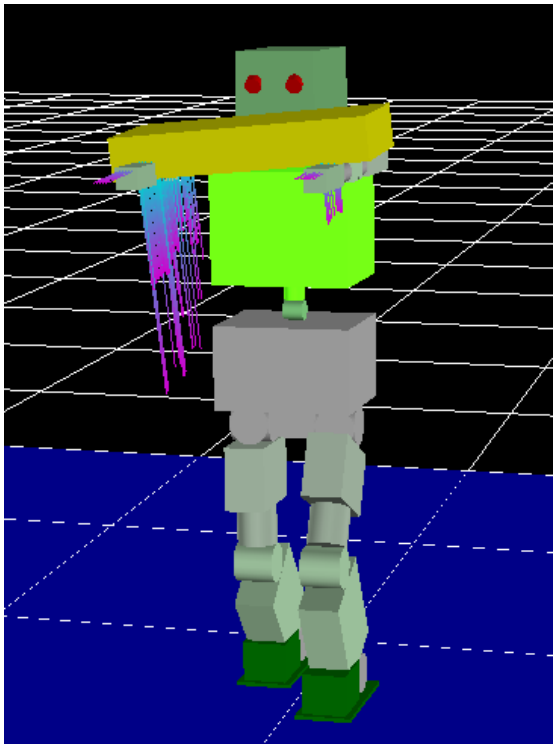

**Figura 5.29 Fin de simulación con 3Kg**

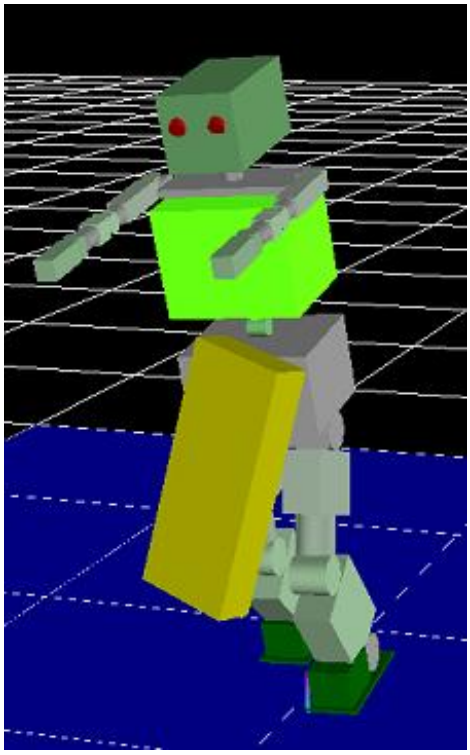

**Figura 5.30 Fin de simulación con 30 Kg**

Cabe destacar que se utilizan estos dos ejemplos debido a que, probando varios pesos, se ha llegado a la conclusión de que el robot se cae al terminar el segundo paso cuando lleva una caja de aproximadamente 25 Kg; por ello, se ha pensado que 3 Kg es un buen peso para demostrar fuerzas con valores de caja comunes en la realidad, y 30 Kg es un valor lo más aproximado a valores reales en los que cae el robot con claridad.

Los grados de libertad referentes al brazo quedan representados en la figura 5.5 (brazo derecho del robot). Como se puede apreciar, son 6 los grados de libertad del brazo ((15), (16), (17), (18), (19) Y (20)) y uno en la pierna: el tobillo representado como un (2).
Es necesario señalar que el número indicado no representa el mismo que el asociado en el archivo VRML del robot.

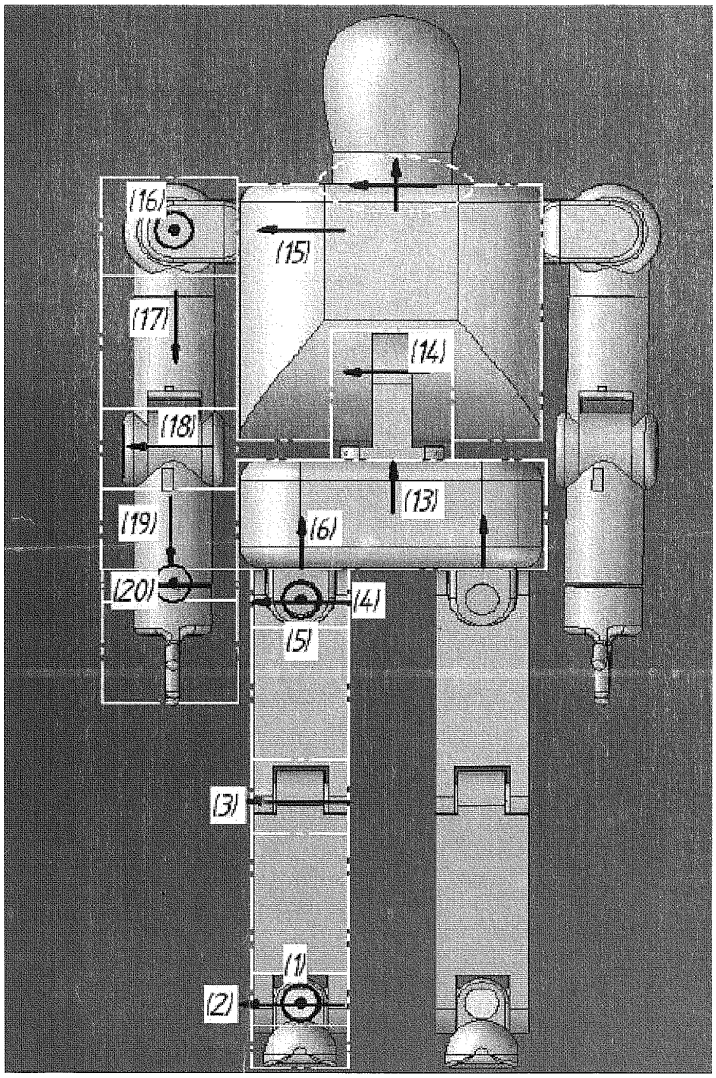

**Figura 5.31 Articulaciones del Rh-2**

Con respecto al brazo, van a ser estudiados los pares en tres articulaciones: hombro, codo y muñeca. En la figura 5.6 quedan indicados en color rojo.

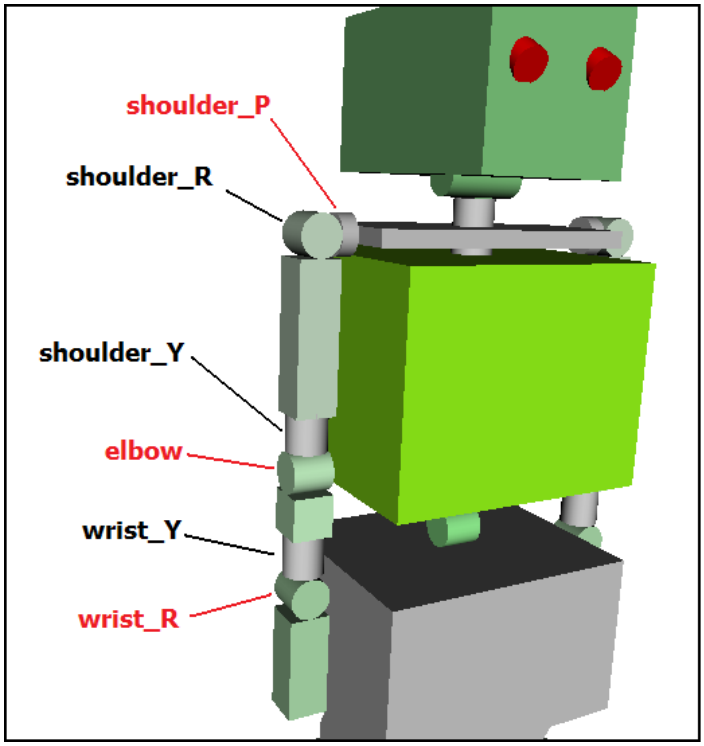


**Figura 5.32 Articulaciones estudiadas**

La caja cae sobre los brazos como indica la figura 5.7, donde el codo está más cercano al centro de gravedad de la caja que la muñeca, la cual no sostiene casi peso.

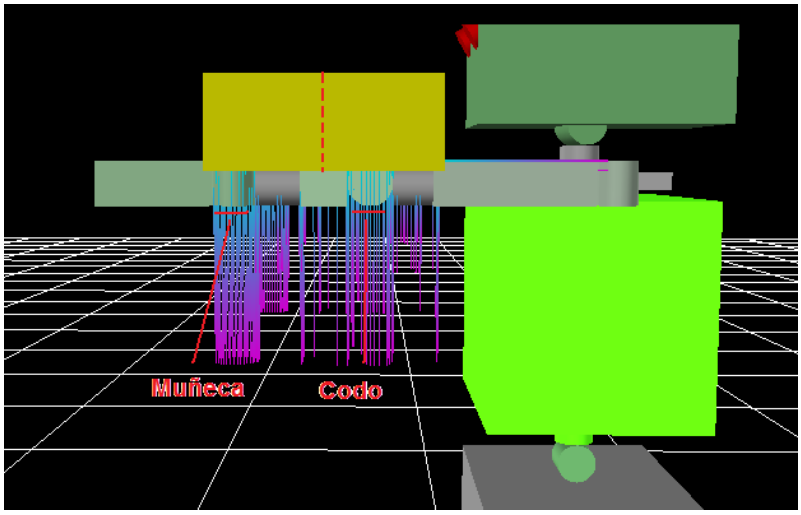


**Figura 5.33 Vista lateral de caja sobre brazos**

Desde la perspectiva de la figura 5.8 se puede apreciar que la articulación del hombro está en distinto plano al del codo y la muñeca.

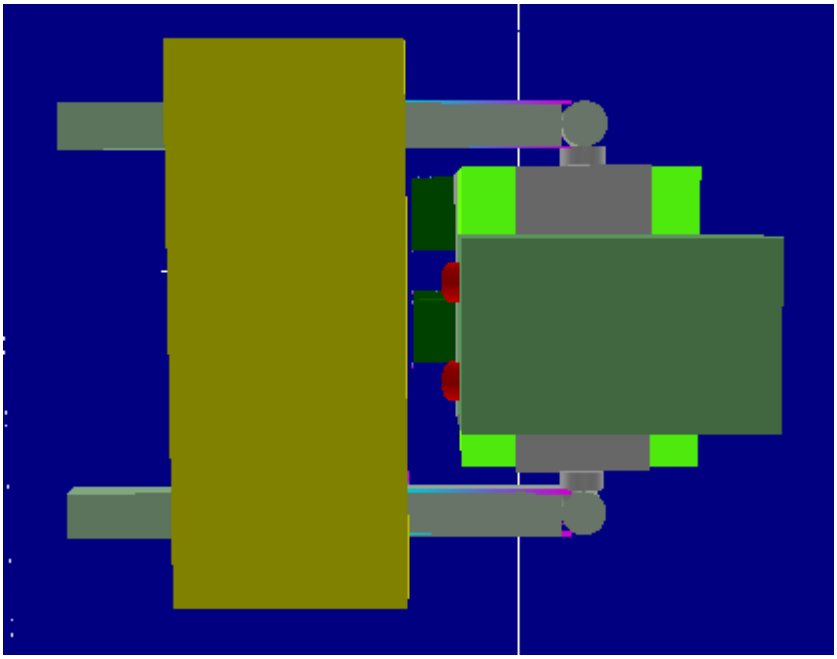

**Figura 5.34 Vista desde arriba de caja sobre brazos**

En la figura 5.9 se muestra la articulación del tobillo. Queda representada en gris.

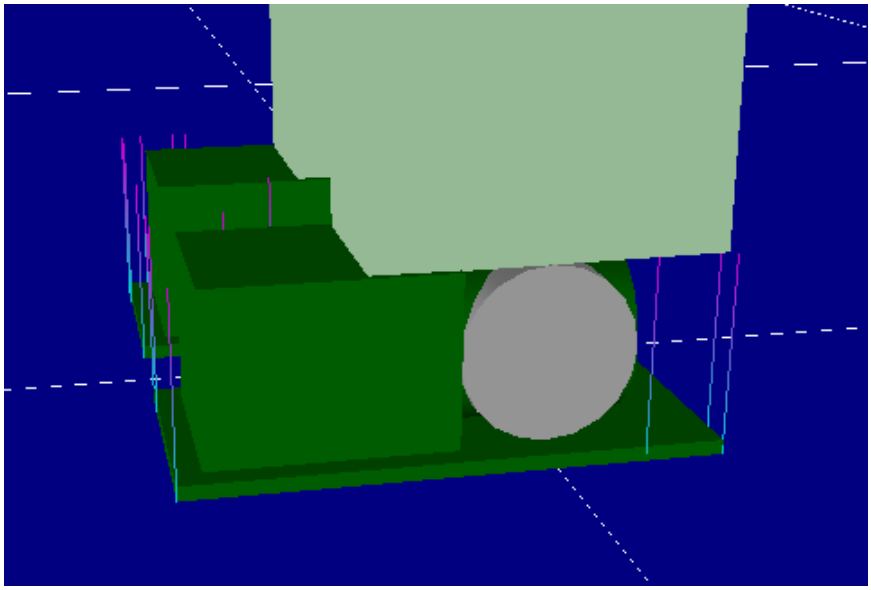

**Figura 5.35 Vista de tobillo**

## 4.2 Modificación de archivos

En primer lugar, se debe modificar el robot, de tal manera que tenga los brazos perpendiculares al tronco y hacia delante, para sostener así la caja. Para ello, se debe modificar los archivos anteriormente nombrados “angle.dat” y “sampleHG.xml”. Este último permite también modificar los parámetros iniciales de la caja, como se explicará más adelante.

En “angle.dat” se modifica la columna equivalente a los hombros dándole el valor de 90º, como se muestra en la figura 5.10.

| G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -7.13E-06 | 0 | 1.43E-05 | -7.13E-06 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -2.84E-05 | 0 | 5.69E-05 | -2.84E-05 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -6.39E-05 | 0 | 0.00012774 | -6.39E-05 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00011332 | 0 | 0.00022664 | -0.00011332 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00017671 | 0 | 0.00035342 | -0.00017671 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00025395 | 0 | 0.0005079 | -0.00025395 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00034495 | 0 | 0.00068991 | -0.00034495 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00044964 | 0 | 0.00089928 | -0.00044964 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00056793 | 0 | 0.00113585 | -0.00056793 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00069972 | 0 | 0.00139945 | -0.00069972 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00084495 | 0 | 0.0016899 | -0.00084495 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00100352 | 0 | 0.00200704 | -0.00100352 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00117535 | 0 | 0.0023507 | -0.00117535 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00136035 | 0 | 0.0027207 | -0.00136035 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00155845 | 0 | 0.00311689 | -0.00155845 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00176955 | 0 | 0.0035391 | -0.00176955 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00199357 | 0 | 0.00398715 | -0.00199357 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00223044 | 0 | 0.00446088 | -0.00223044 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00248006 | 0 | 0.00496013 | -0.00248006 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00274236 | 0 | 0.00548472 | -0.00274236 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00301724 | 0 | 0.00603449 | -0.00301724 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00330464 | 0 | 0.00660928 | -0.00330464 | 0 | -1.57079633 |
| 0 | -1.57079633 | 0 | 0 | 0 | 0 | 0 | 0 | -0.00360446 | 0 | 0.00720891 | -0.00360446 | 0 | -1.57079633 |

**Figura 5.36 modificación columna datos de hombros.**

De este modo, el robot mantendrá los brazos en esta posición durante toda su simulación. La columna del archivo “acc.dat” y “vel.dat” equivalente a dichos hombros se mantendrá a cero, dado que permanecen en posición constante.

En el archivo “sampleHG.xml” se configuran las posiciones iniciales del robot, así como las de la caja. Es importante centrar la caja en los brazos dándole el valor cero respecto al eje “y”

Tal y como se muestra en la figura 5.11 los valores iniciales de coordenadas de la caja están configurados para que ésta caiga a la altura de los codos. Debido a que la caja es un joint de tipo “free”, le permite moverse libremente e interaccionar con otro objeto. La altura de la caja está ligeramente por encima de la posición de los brazos, con el objetivo de que caiga sobre ellos al principio de la simulación.

```
<item class="com.generalrobotix.ui.item.GrxModelItem" name="box3" select="true" url="$(OPENHRPHOME)/etc/box3.wrl">
    <property name="isRobot" value="false"/>
    <property name="WAIST.rotation" value="0.0 0.0 0.0 0.0 "/>
    <property name="WAIST.translation" value="0.32 0.0 1.65 "/>
</item>
```

**Figura 5.37 valores iniciales de la caja**

En este archivo también se configura que haya interacción entre la caja y el robot, tal y como muestra la figura 5.12.

```
<item class="com.generalrobotix.ui.item.GrxCollisionPairItem" name="CP#box3#sample" select="true">
    <property name="springConstant" value="0 0 0 0 0 0"/>
    <property name="slidingFriction" value="0.5"/>
    <property name="jointName2" value=""/>
    <property name="jointName1" value=""/>
    <property name="sprintDamperModel" value="false"/>
    <property name="damperConstant" value="0 0 0 0 0 0"/>
    <property name="objectName2" value="sample"/>
    <property name="objectName1" value="box3"/>
    <property name="staticFriction" value="0.5"/>
</item>
```

**Figura 5.38 Interacción entre caja y robot.**

En segundo lugar, y no menos importante, se debe modificar el archivo “box.wrl”, dando a la caja su tamaño correcto (para que los dos brazos la sostengan) y su masa, que irá siendo modificada para comprobar las variaciones de fuerzas. Lo que se muestra en la figura 5.13 es la parte del archivo que se modifica.

```
DEF WAIST Joint {
  jointType "free"
  translation 0.55 -0.02 0.15
  rotation 0 1 0 0.2
  children [
    DEF BODY Segment {
  #====== wood block (cedar tree) ============
  mass 3
  momentsOfInertia [0.01 0 0 0 0.001 0 0 0 0.01]
  #====== aluminum block ==============
  #mass 86.4
  #momentsOfInertia [4.896 0 0 0 0.57599 0 0 0 4.896]
      children Transform {
    translation 0 0 0
    rotation 1 0 0 0
        children Shape {
          geometry Box {
              size 0.3 0.7 0.1              }
          appearance Appearance {
```

**Figura 5.39 Modificación masa de caja en VRML.**

## 4.3 Iniciación de la simulación

Una vez los archivos están listos para la simulación, se carga el proyecto en el simulador GrxUI como se ha explicado con anterioridad.
Inicialmente, al ser cargado el proyecto, la caja aparece más arriba de los brazos, como refleja la figura 5.14.

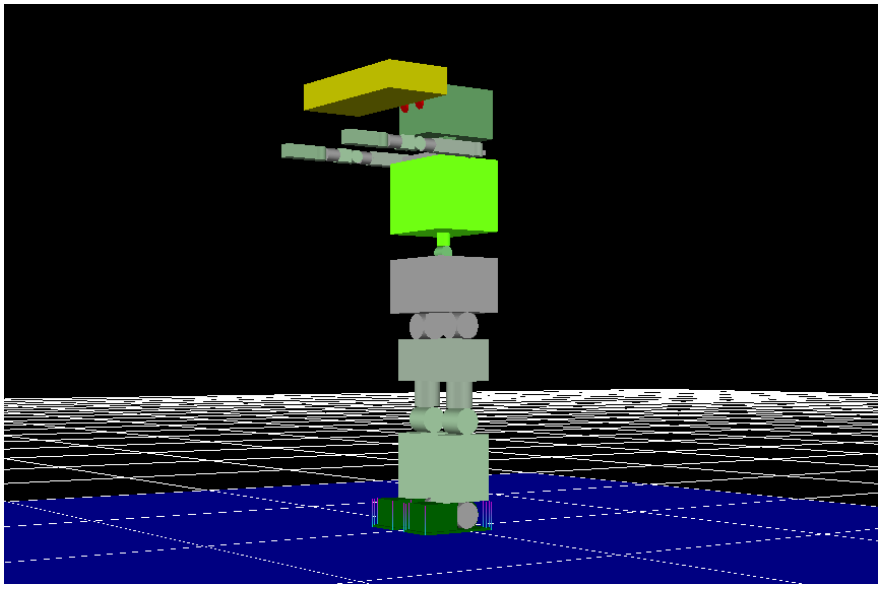

**Figura 5.40 Posición inicial.**

Iniciada la simulación, la caja cae sobre los brazos, tras aplicarse la fuerza de la gravedad en ella. Es en este instante cuando la caja empieza a interaccionar con el robot, el cual comienza a dar el paso. Esto es un punto importante representado más adelante.

## 4.4 Momentos importantes del paso

La simulación del paso con la caja dura siete segundos, y consta de varios puntos críticos, los cuales hacen que la gráfica tenga cambios considerables.
Estos puntos, que serán enumerados a continuación, se reflejan con una imagen del robot en plena simulación y la parte que representa en las gráficas (para esto se tomará la gráfica del sensor situado en el hombro izquierdo con una caja de 3 Kg, para los tres ejes).
La figura 5.15 resume claramente las fases del paso en la gráfica, para mayor claridad.

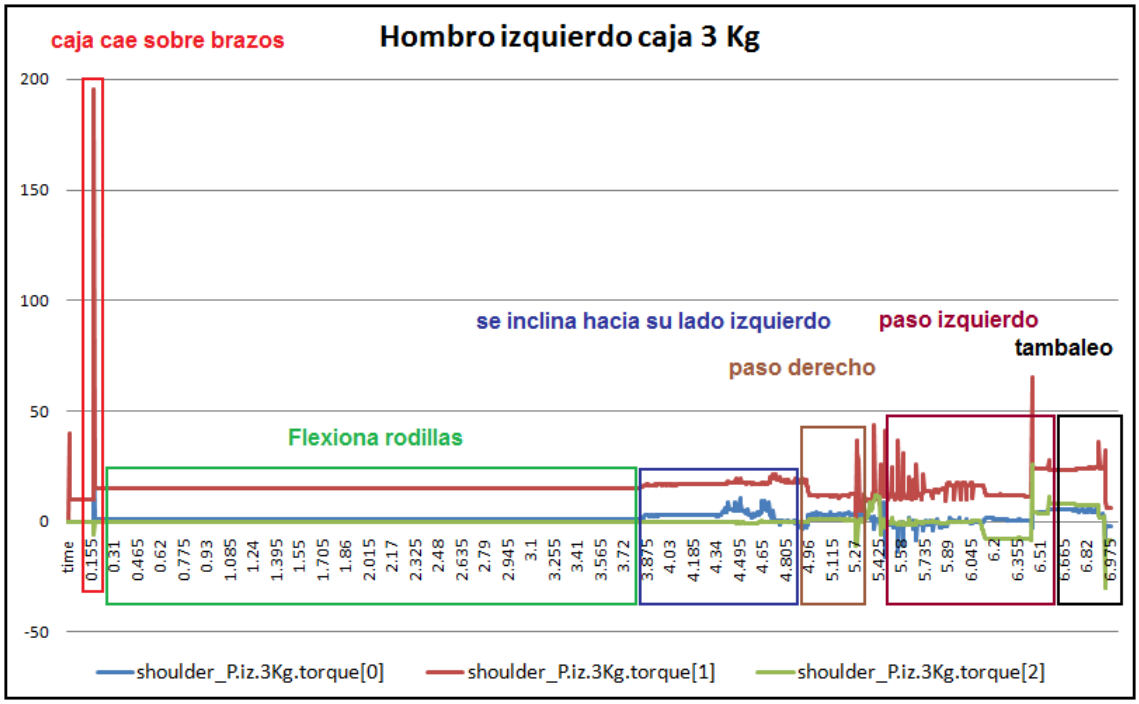


**Figura 5.41 Resumen fases del paso**

### 4.4.1 Colisión caja con brazos

Al iniciar la simulación, la caja está situada por encima de los brazos y cae debido a la gravedad, alcanzando a los mismos en el segundo 0.180. Ver figura 5.16.

En la gráfica de la figura 5.17 se aprecia un pico de sobrecarga, ya que la caja cae de golpe.

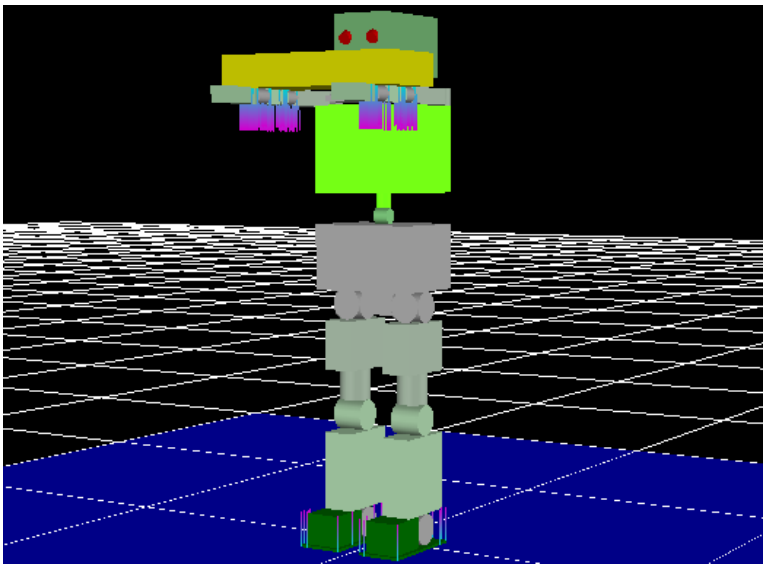

**Figura 5.42 Caja cae sobre brazos**

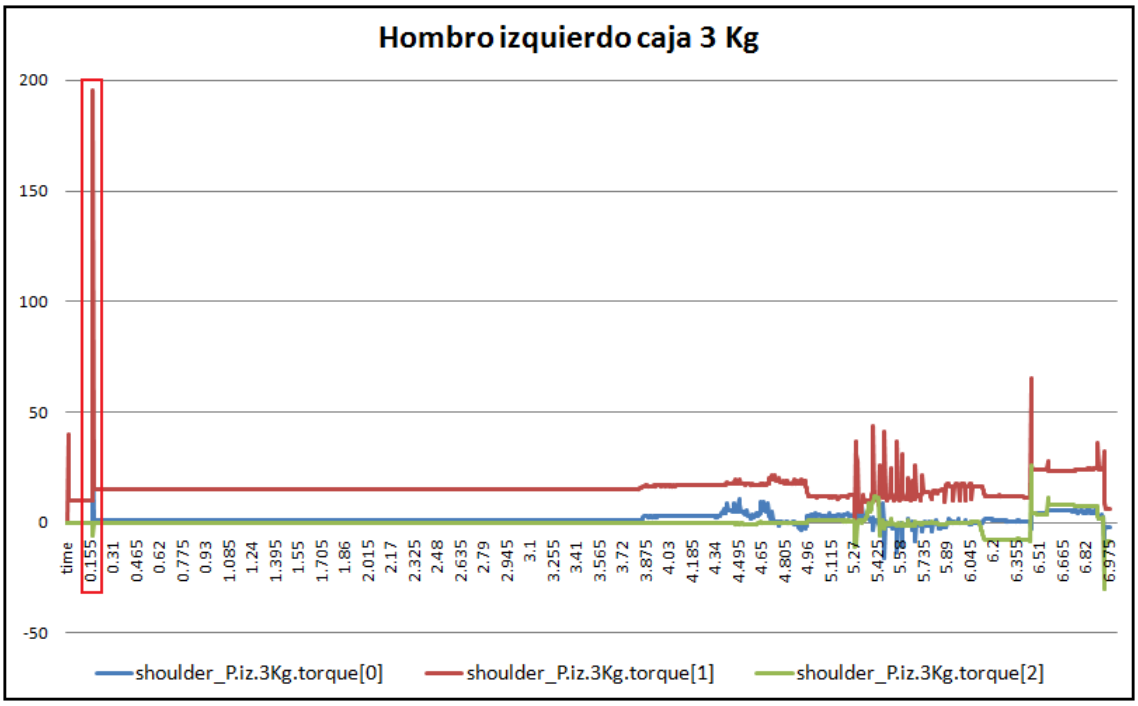


**Figura 5.43 Gráfico caja cae sobre brazos**

## 4.4.2 Robot empieza a inclinarse

El robot comienza a inclinarse hacia su lado izquierdo y la caja empieza a "introducirse" dentro del brazo derecho, por el error del simulador comentado anteriormente. Esto se muestra en la figura 5.18 y sucede en el segundo 3.910.

En la gráfica de la figura 5.19 se evidencia una distorsión en la regularidad que tenía durante la flexión de las rodillas.

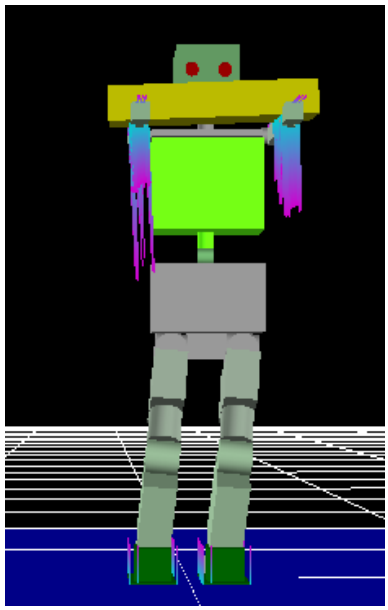

**Figura 5.44 Robot comienza a inclinarse**

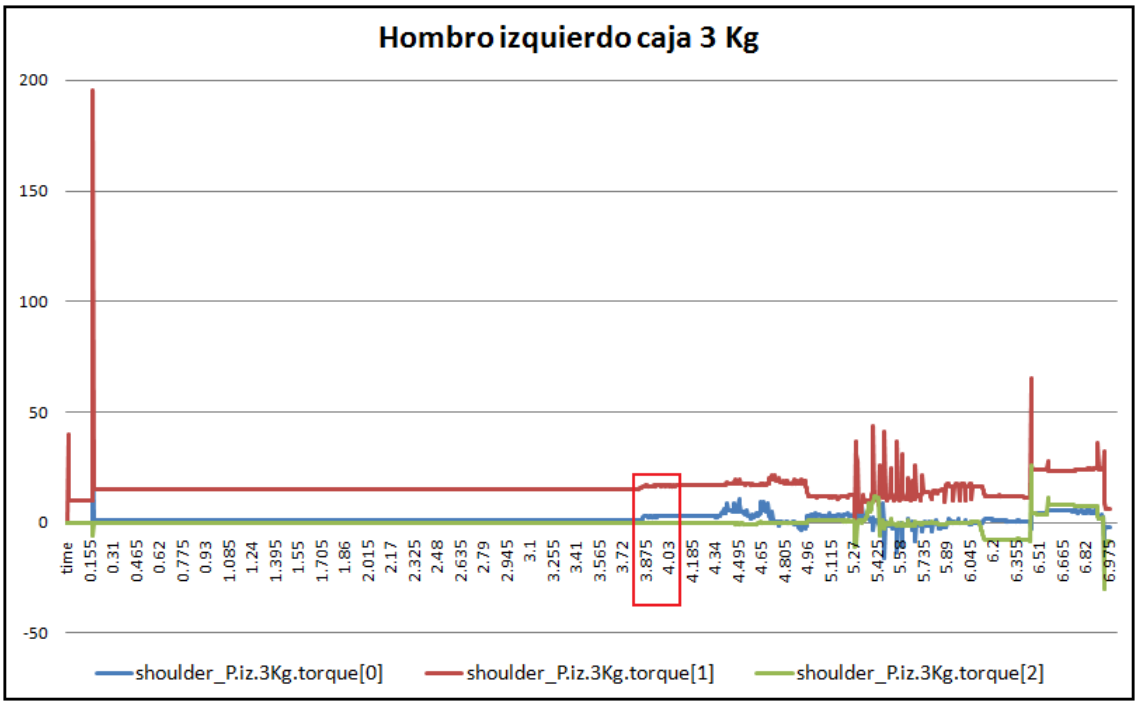


**Figura 5.45 Gráfico robot comienza a inclinarse**

### 4.4.3 El pie derecho se levanta para dar el paso

En el segundo 4.860, el pie derecho se eleva del suelo para dar el primer paso. En la figura 5.20, se puede advertir que la caja está totalmente insertada en el brazo derecho.

Gráficamente hablando, éste no es un punto crítico, tal y como se divisa en la figura 5.21, aunque las líneas de contacto se disparen.

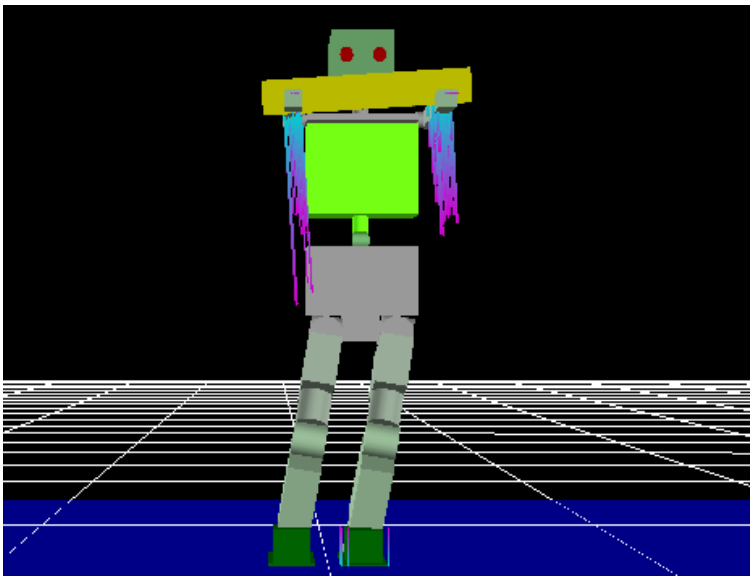

**Figura 5.46 Pie derecho comienza a levantarse**

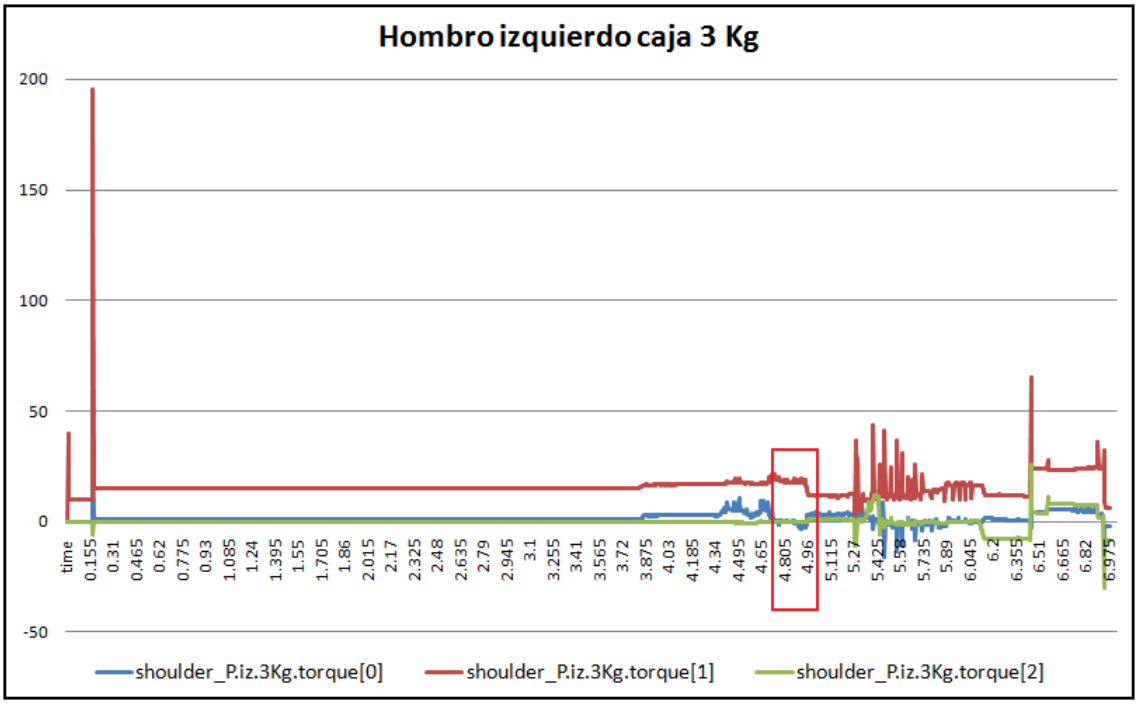


**Figura 5.47 Gráfico pie derecho comienza a levantarse**

### 4.4.4 Pisa con el pie derecho

Después de dar el primer paso, el robot continúa hacia adelante, apoyando el pie derecho, como se puede apreciar en la figura 5.22. El segundo que indica el simulador es el 5.290.

Gráficamente, en la figura 5.23 se percibe un pico en el momento de la pisada.

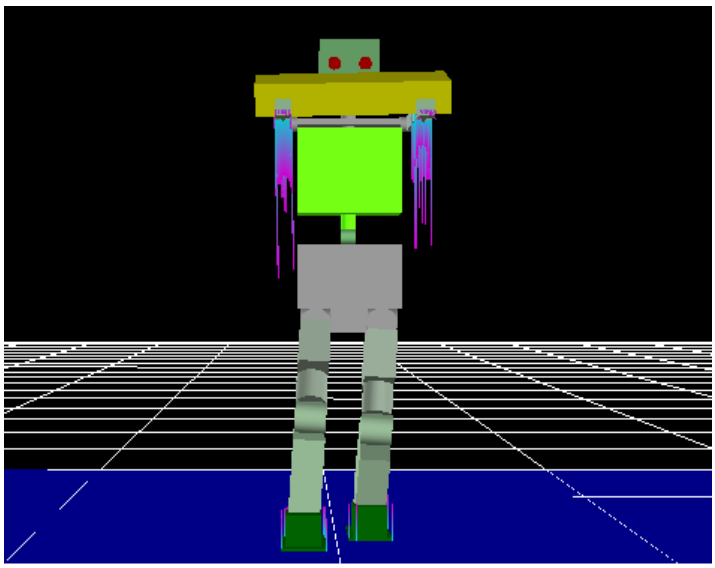

**Figura 5.48 Término del primer paso**

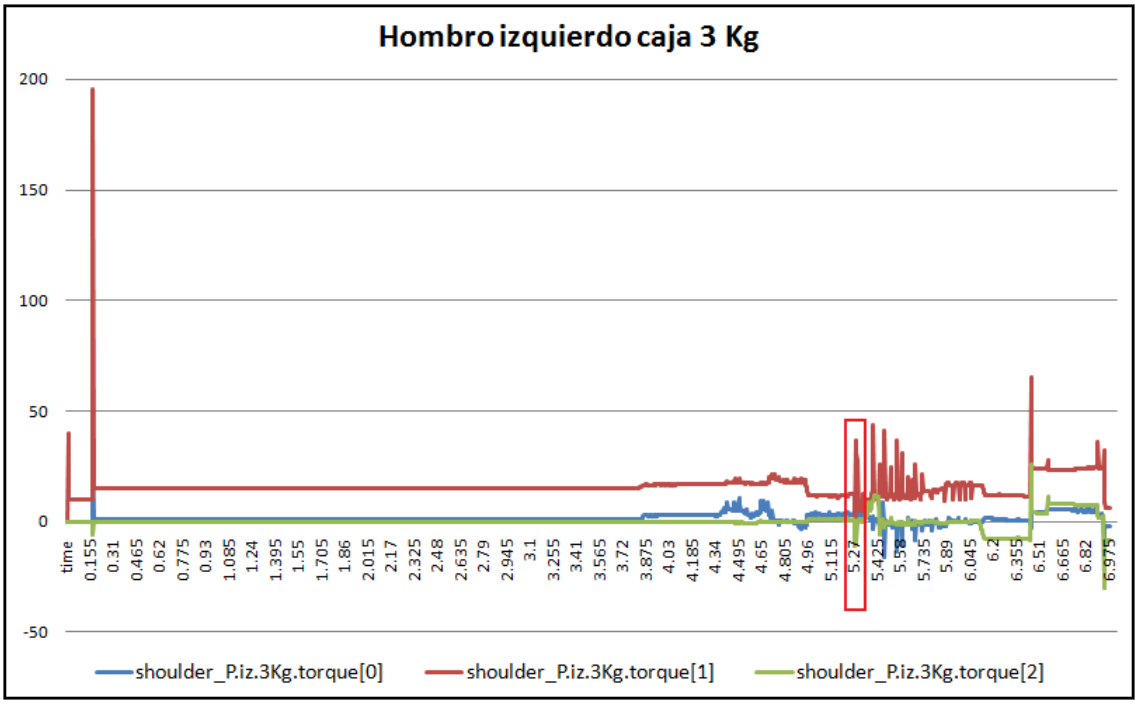


**Figura 5.49 Gráfico término del primer paso**

### 4.4.5 Robot levanta pie izquierdo

Cuando se dispone a dar el segundo paso, el pie izquierdo deja de estar en contacto con el suelo. El segundo indicado es el 5.500. Ver figura 5.24.

Se puede distinguir en la gráfica de la figura 5.25 que en el transcurso de levantar el pie izquierdo existen unos valores de par dados en picos.

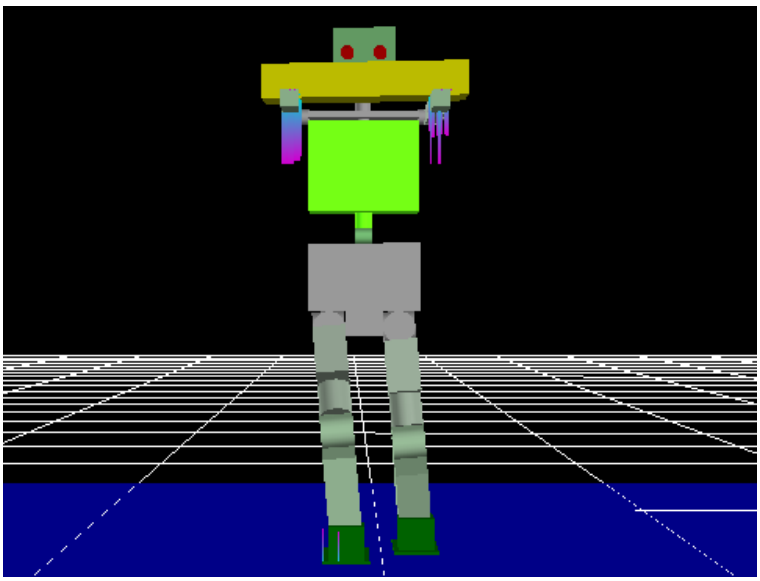

**Figura 5.50 Comienzo del segundo paso.**

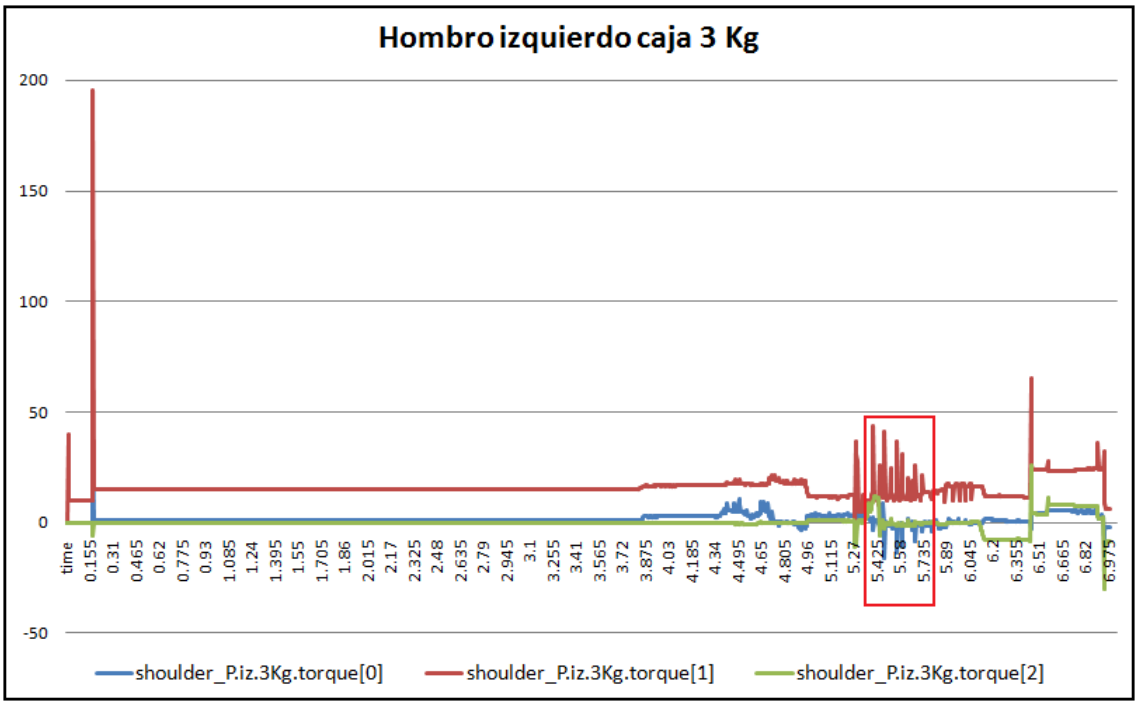


**Figura 5.51 Gráfico comienzo del segundo paso**

### 4.4.6 Robot pisa con pie izquierdo

El robot termina el segundo paso situando su pie izquierdo sobre el suelo. Esto sucede casi al final, en el segundo 6.585, como se advierte en la figura 5.26.

En la gráfica se aprecia claramente el momento del fin del paso, con un pico claro (figura 5.27).

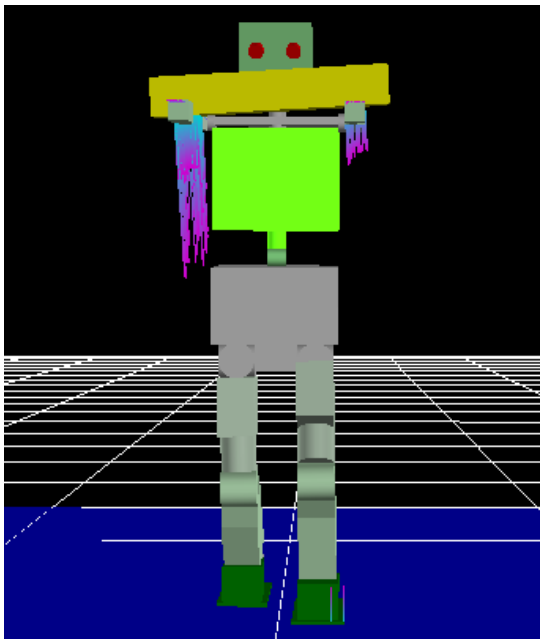

**Figura 5.52 Fin del segundo paso**

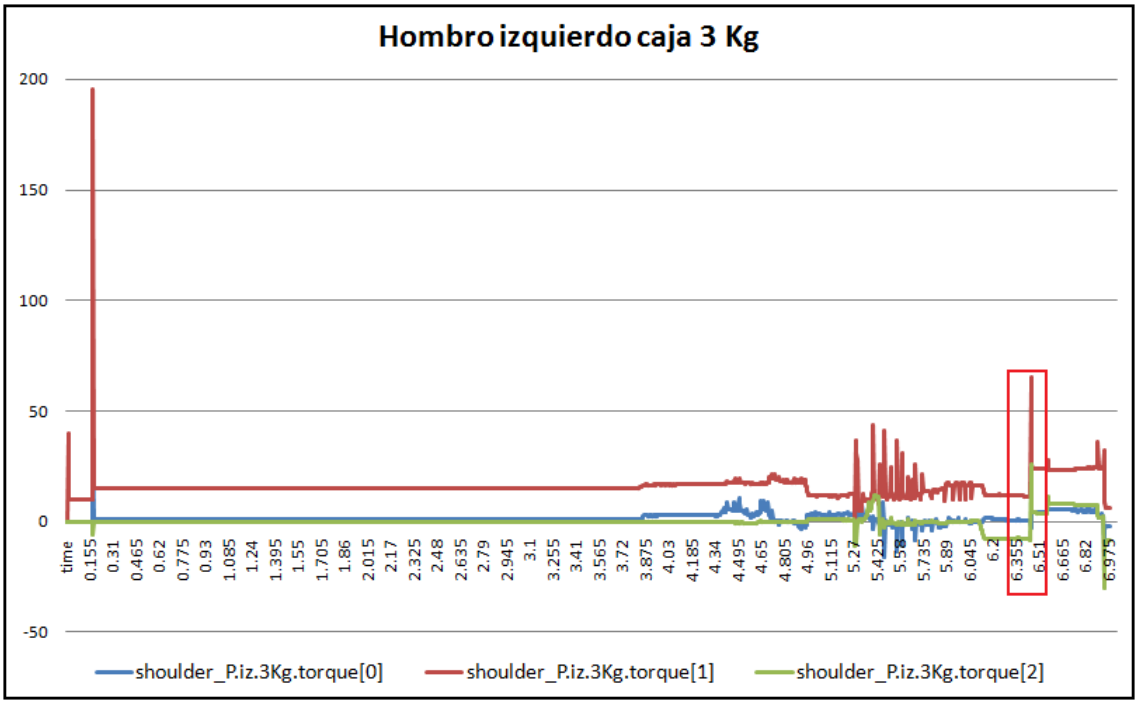


**Figura 5.53 Gráfico fin del paso**

### 4.4.7 Tambaleo del robot tras finalizar el paso

Tras finalizar el paso, el robot queda en equilibrio presentando un ligero tambaleo centrado primariamente en el pie recién apoyado (pie izquierdo). Se puede percibir el apoyo del p8ie sobre la punta del pie. Después de esto, el robot se balancea hasta apoyarse en el derecho. (Ver figura 5.28).

Gráficamente se observa un cambio considerable de continuidad después de terminar el paso, aproximadamente en el segundo 6.975. Esto se debe al tambaleo, en el que el hombro siente un sobreesfuerzo. (Ver figura 5.29).

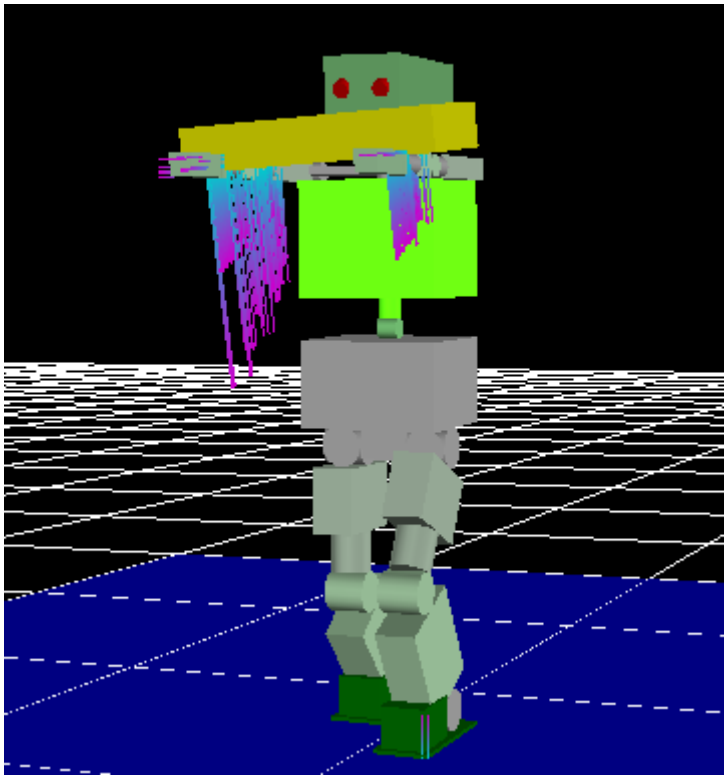

**Figura 5.54 tambaleo**

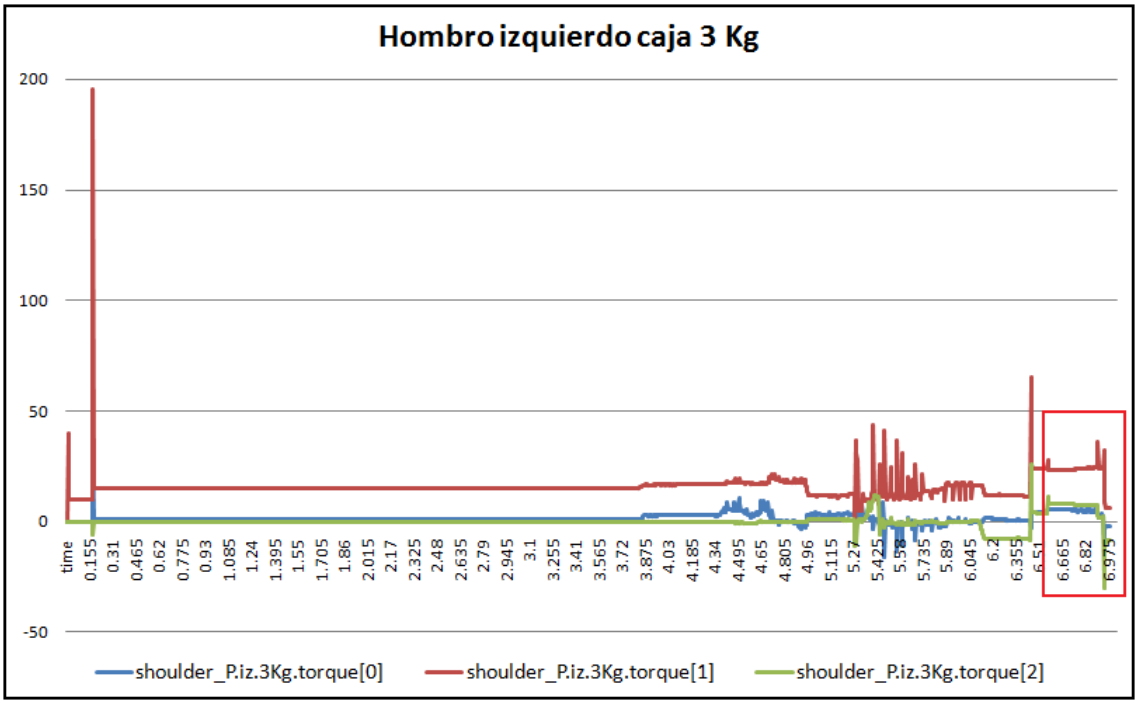


**Figura 5.55 Gráfico tambaleo**

# 4.5 Estudio del paso sin caja, con 3 Kg y 30 Kg.

Este apartado va a recoger los valores de los sensores de hombros, codos, muñecas y tobillos en sus distintos ejes. Se examinarán los distintos ejes de cada articulación mediante gráficas individuales, donde quede reflejado los diferentes resultados, tanto los del robot libre como los obtenidos tras la incorporación de las cajas de peso (3 Kg, y 30 Kg).

En el caso de los brazos (hombros, codos y muñecas), se representará solamente la parte izquierda del robot, ya que tanto la parte izquierda como la derecha dan resultados semejantes, siendo la parte izquierda algo más alta en cuanto a valores. Esto queda demostrado en las figuras 5.30 y 5.31.

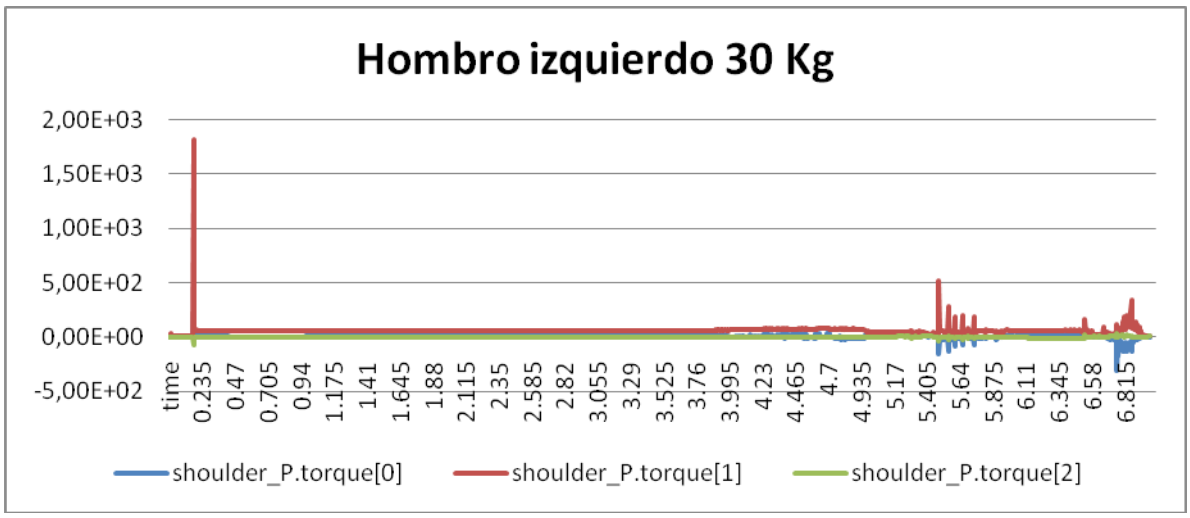


**Figura 5.56 Hombro izquierdo con 30 Kg**

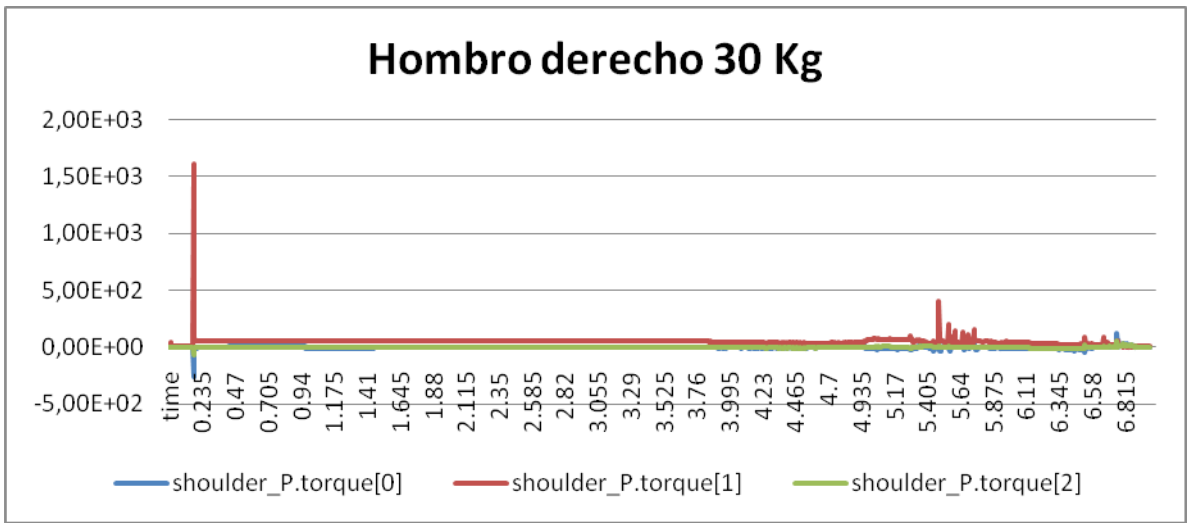


**Figura 5.57 Hombro derecho con 30 Kg**

### 4.5.1 Hombro

Hay que destacar que, como se ha mostrado en imágenes anteriores, el hombro está en otro plano XZ con respecto al del brazo. Esto hace que los momentos con respecto al eje “x” sean mayores a los de los codos y muñecas.

#### 4.5.1.1 Eje X

En la gráfica de la figura 5.32 se observa que el robot no siente prácticamente par en el caso de sin caja y con la caja de 3 Kg, ya que en el eje x el par que siente está dado por la diferencia de distancia entre el plano del hombro y el del codo y la muñeca.

Hay un pico, de valor 270 N*m, en el momento en que la caja de 30 Kg cae sobre los brazos (éste ha sido cortado ya que evitaba ver con claridad los demás valores).

Cuando el robot comienza a inclinarse, se empieza a notar variación en los datos, habiendo un incremento de las fuerzas.

Se observa como los valores más significativos son cuando da el medio paso (pisa pie derecho) y cuando da el paso final, y que estos son negativos, ya que la distancia del hombro izquierdo al plano del centro de la caja es negativa. Este valor tan grande al final se debe a que la caja se cae de los brazos y provoca un sobreesfuerzo de 310N*m negativos.

Se diferencia como en el primer paso, los valores son positivos, dado que la caja está más apoyada sobre el brazo derecho, y se sigue el convenio de la mano derecha. Lo mismo pasa para el segundo paso con valores negativos.

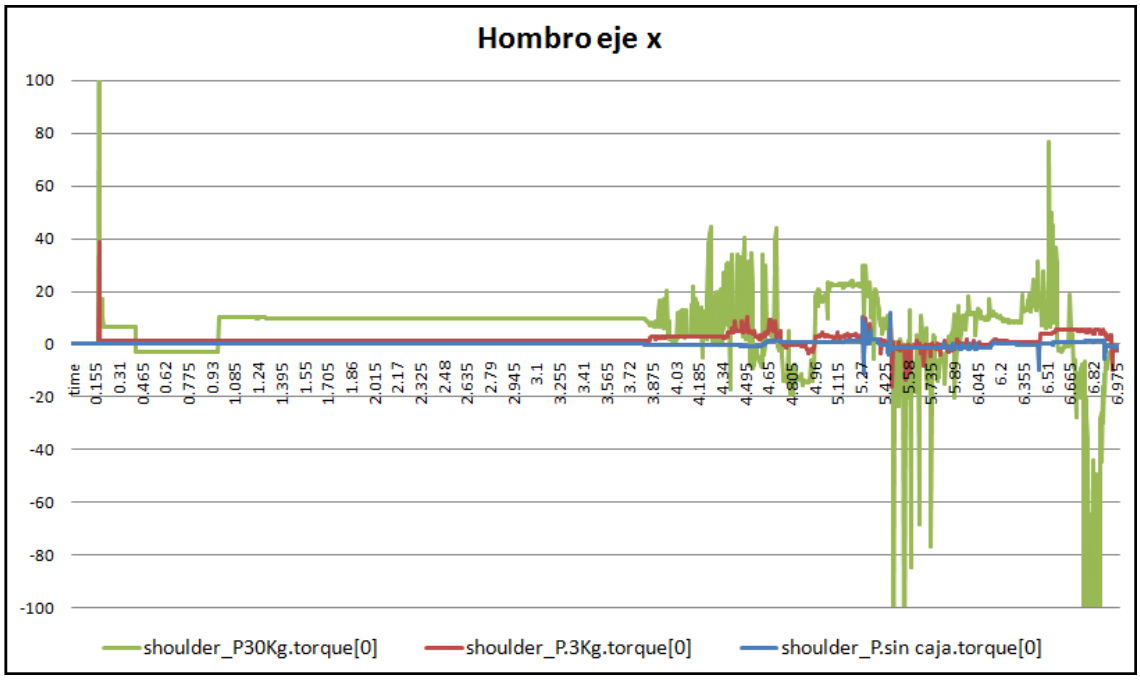

**Figura 5.58 Hombro izquierdo eje X**

#### 4.5.1.2 Eje Y

En el eje “y” es donde se obtienen los valores más grandes, dado que este eje es el más apropiado para sentir el peso de la caja, vertical debido a la fuerza de la gravedad.

Como es lógico y visible en la figura 5.33, el mayor pico se alcanza al caer la caja de 30 Kg sobre los brazos y los otros dos valores considerables son al pisar cada pie en el medio paso y el paso completo.

Se puede apreciar como el par al caer la caja de 3 Kg es de aproximadamente 200 N*m y, a su vez, con la caja de 30 Kg es de unos 1800 N*m (no representado). Con esto se demuestra que se cumple que es unas 10 veces mayor, aunque con un pequeño error. Para verlo mejor, está también el pico del segundo 5.5, donde se levanta el pie izquierdo, de valor 500 N*m con caja de 30 Kg, donde con la caja de 3 Kg se alcanza 50 N*m.

Se observa como, para los casos de sin caja y 3Kg, los valores son semejantes, ya que la caja de poco peso no modifica en gran cantidad el par que sienten los hombros al sostener el peso de los propios brazos (6 Kg). Los valores más significativos que se alcanzan en estos dos casos, son al pisar el pie derecho y el pie izquierdo.

Es importante descatar cómo todos los valores son positivos, por el convenio de la mano derecha, ya que la fuerza ejercida en este caso es el peso de la caja.

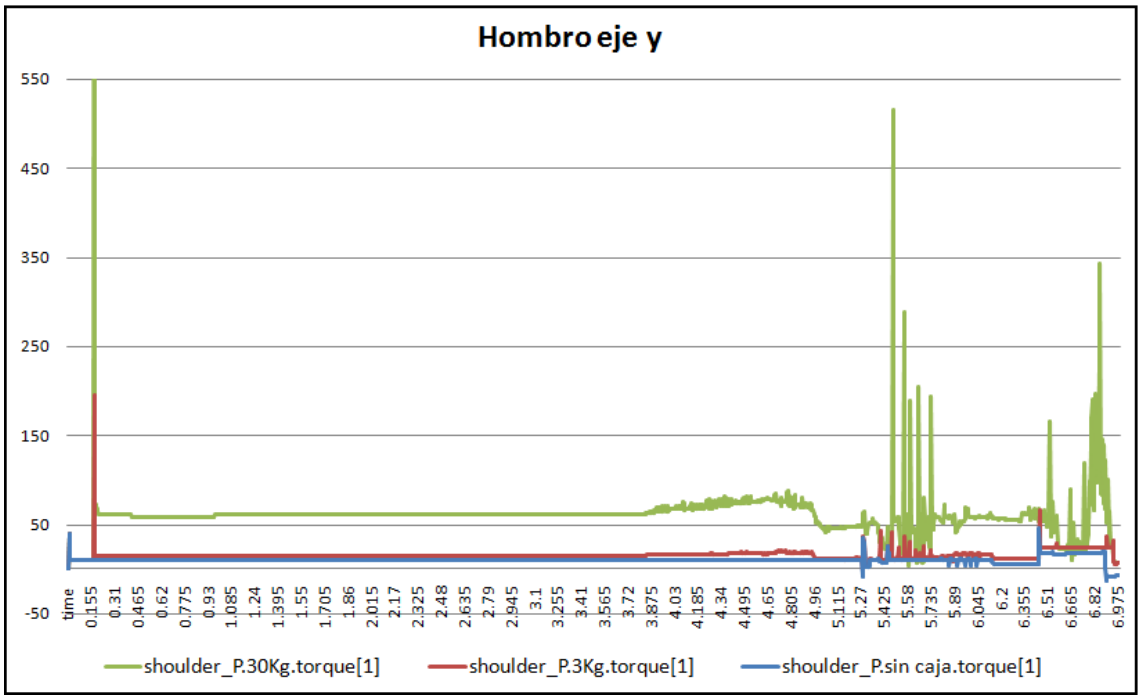


**Figura 5.59 Hombro izquierdo eje Y**

### 4.5.1.3 Eje Z

En el eje “z”, es en el que menos fuerzas siente el robot, ya que al ser un eje vertical en los hombros, sólo sentirá pares con las fuerzas laterales. Teóricamente, estando en reposo hasta el segundo 3.9 (donde empieza a inclinarse), el par debería ser nulo. El pico negativo que aparece al caer la caja es debido a la diferencia de planos entre el brazo y el hombro, sintiendo el hombro lateralmente parte de esa fuerza hacia abajo, ya que la caja cae verticalmente (ver figura 5.34). Este pico alcanza los 70 N*m.

Una vez comenzada la inclinación lateral para empezar a dar el paso, sí es lógico que empiecen a aparecer los pares, ya que el robot irá de balanceándose de lado a lado en pleno paso. Estos pares tienen sus máximos en el momento de poner el pie derecho (medio paso) y en el término del mismo (pisar pie izquierdo).

Se observa también, que el caso de 30 Kg es distinto a los de 3 Kg y el de sin caja, ya que al término del paso, los picos con la caja de 3 Kg y sin la caja son negativos. Esto se debe a la desestabilización de la finalización del paso cuando cae la caja de los brazos.

Se observa que los casos de 3 Kg y sin caja son semejantes, ya que al no haber prácticamente fuerzas laterales, pasar de 0 a 3 Kg es inapreciable. La mayoría de estas fuerzas son debidas al brazo en sí, que está en otro plano al del hombro.

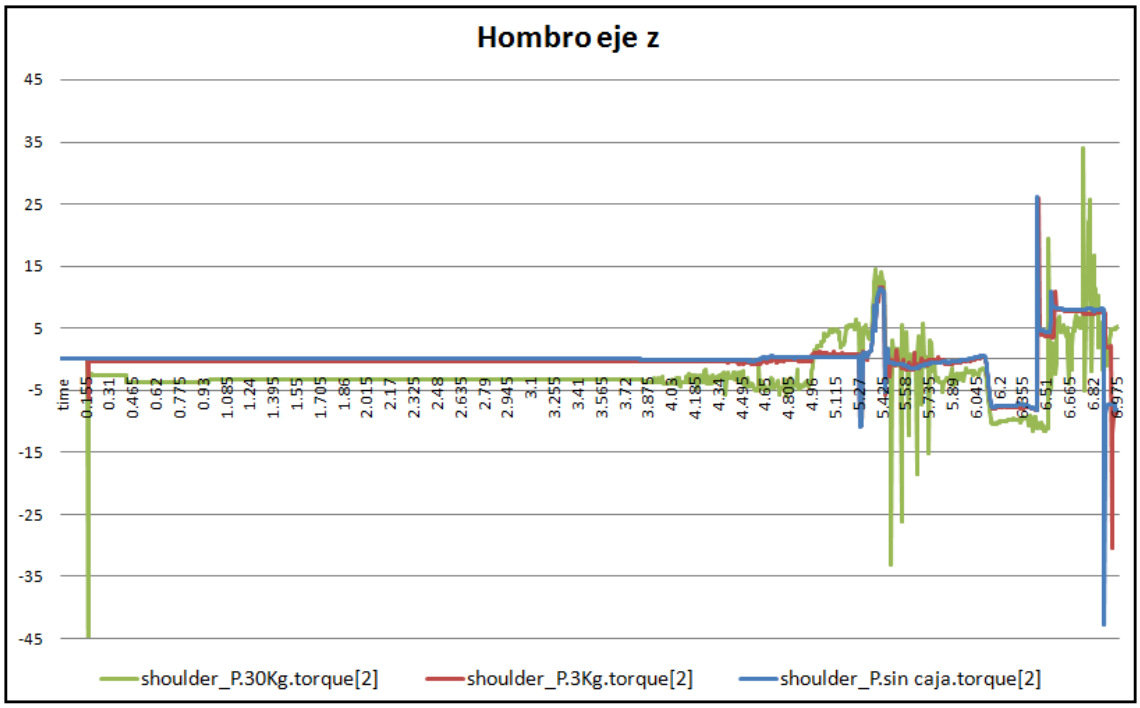


**Figura 5.60 Hombro izquierdo eje Z**

## 4.5.2 Codo

Con respecto a la caja, los codos están situados prácticamente en el medio de ésta, ligeramente más arriba, hacia el hombro, y en el mismo plano que las muñecas. Es una de las articulaciones que sostiene la caja y los pares, tanto en el izquierdo como en el derecho, son parecidos, exceptuando el momento del paso, puesto que al elevar el pie derecho, el robot está inclinado hacia la izquierda, y el codo izquierdo siente más par que el derecho, y viceversa con el paso del pie izquierdo.

### 4.5.2.1 Eje X

En el eje “x”, los pares que sienten los sensores son inferiores a los del hombro, ya que están en el mismo plano del apoyo de la caja. Aún así, sienten fuerzas, las cuales son mucho mayores que en el caso de la caja de 30 Kg.

En esta situación, existe algo que destaca en la figura 5.35, y es que el mayor par que siente el codo respecto a su eje “x”, es en el momento de finalizar el paso, y éste es de valor 90 N*m negativos. Este incremento tan grande se debe a la caída de la caja.

Otro valor máximo que aparece es al caer la caja de 30 Kg sobre los brazos, el cual alcanza los 60 N*m.

Se contempla como en el caso de sin caja, sólo hay 3 picos, siendo el resto de valores prácticamente cero, exceptuando el momento en el que el robot está dando el primer

paso. Estos picos se alcanzan en el momento de terminar el primer paso con el pie derecho, al levantar el pie izquierdo y al finalizar el paso.

En este caso pasa lo mismo con los signos de los pares que lo explicado en el caso del eje “x” los hombros.

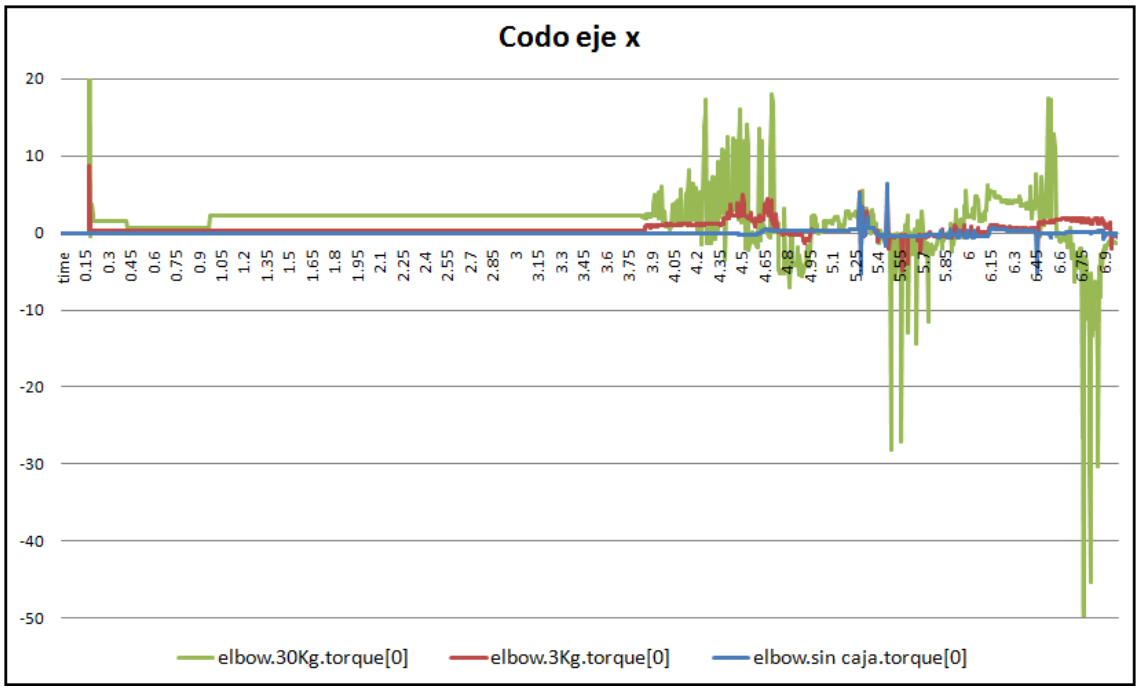


**Figura 5.61 Codo izquierdo eje X**

### 4.5.2.2 Eje Y

En este caso, también son inferiores los valores comparados con el hombro, ya que respecto al eje “y”, la distancia es menor, dado que la caja cae sobre los brazos.

El mayor pico es en el momento de caer la caja, el cual alcanza un valor de 350 N*m. Además, se resalta un incremento según se va inclinando el robot a la izquierda hasta que eleva el pie derecho (ver figura 5.36), dado que estamos estudiando el codo izquierdo (en el caso del codo derecho el valor disminuye hacia cero). Después, desde que levanta el pie derecho hasta que lo apoya en el suelo, el valor es aproximadamente cero, dado que en ese intervalo de tiempo la caja se apoya mayoritariamente sobre el brazo derecho (como se mostró antes, es el momento en el que la caja se fusiona con el brazo derecho). Mientras ejecuta el paso izquierdo, los valores transcurren con normalidad, excepto al terminar, que existe un pico de valor negativo, el cual no se puede explicar de manera teórica, puesto que la caja tiene un

peso hacia abajo, aunque se excusa con la figura 5.37, en la que el simulador toma el contacto entre la caja y el brazo desde abajo.

La mayoría de valores son positivos de acuerdo con la regla de la mano derecha.

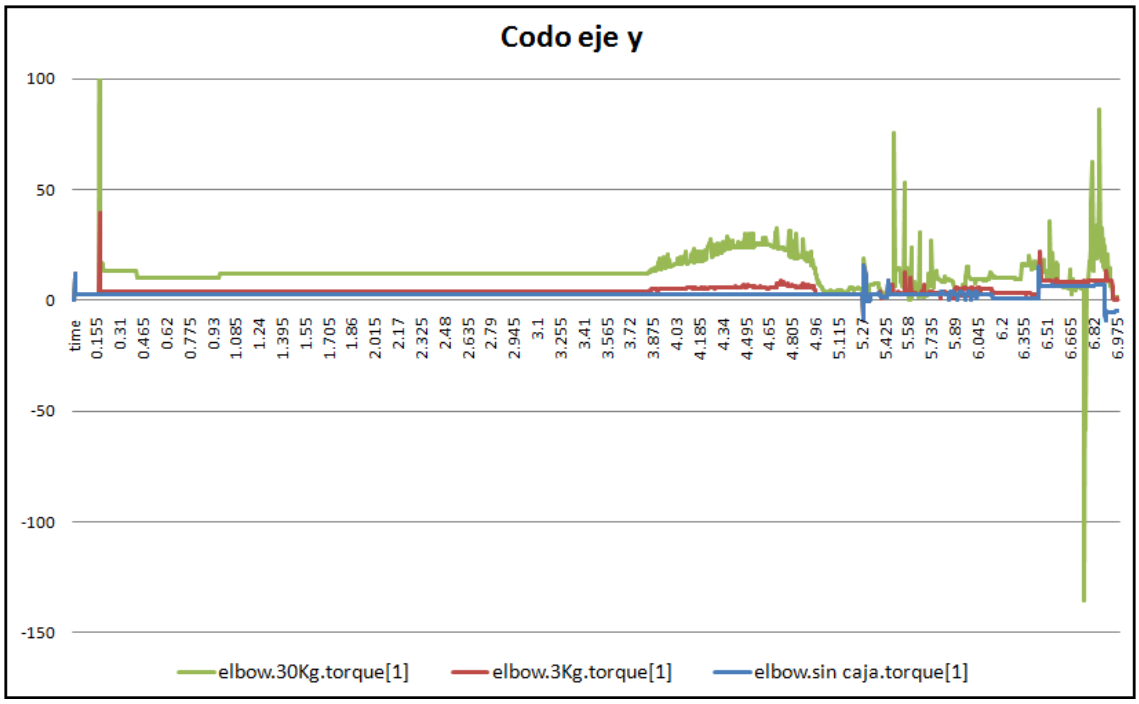


**Figura 5.62 Codo izquierdo eje Y**

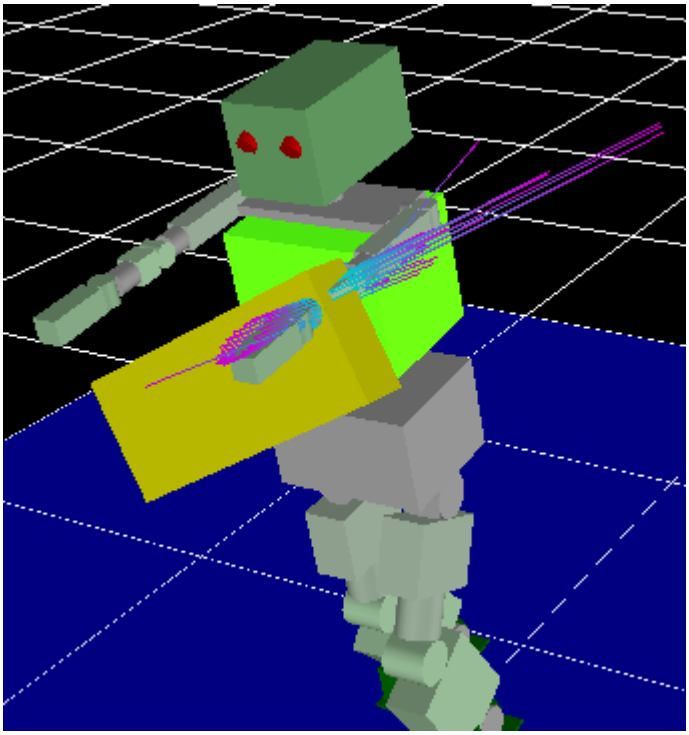

**Figura 5.63 Caja cae y atraviesa el brazo**

#### 4.5.2.3 Eje Z

Este caso es similar al del eje “z” del hombro, con valores muy semejantes.


**Figura 5.64 Codo izquierdo eje Z**

### 4.5.3 Muñeca

La parte de la muñeca es la más simple de estudiar, ya que está situada debajo de la caja pero más adelante, hacia el final de los brazos, y por este motivo no tiene la función de sostener la caja ni de hacer fuerza para ello.

#### 4.5.3.1 Eje X

Con respecto al eje “x”, los pares son muy pequeños, incluso con la caja de 30 Kg, ya que las fuerzas verticales ejercidas sobre la muñeca tienen poco recorrido (la distancia a los sensores es mínima).

Como se puede ver en la figura 5.39, hay un pico considerable en el momento de finalizar el paso, cuando la caja cae y atraviesa el brazo izquierdo como se mostró con anterioridad. Los datos en este momento no son del todo fiables.

La parte más regular de valores se halla en el transcurso de la inclinación del robot, tanto para 3 Kg como para 30Kg.

Los valores sin caja son iguales que anteriormente, reflejando los valores máximos al posar el pie derecho, elevar el pie izquierdo y finalizar el paso del mismo.

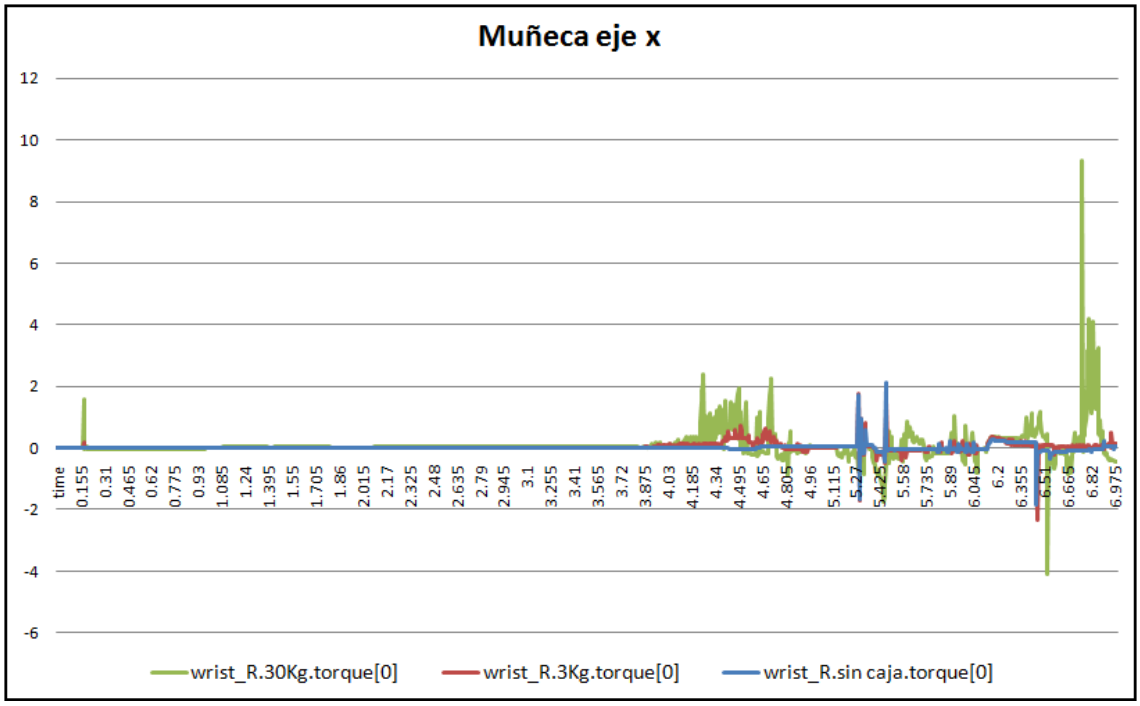


**Figura 5.65 Muñeca izquierda eje X**

### 4.5.3.2 Eje Y

Observando el comportamiento de los sensores con respecto al eje “y”, se puede contemplar que los datos son igual de pequeños, cuando deberían ser algo más grandes. (Ver figura 5.40). Esto puede deberse a lo nombrado al principio de este subtema de las muñecas, o a lo mostrado en la figura 5.41, donde se aprecia una inclinación de la caja de 30 Kg sobre las muñecas, disminuyendo como consecuencia el par generado.

Al final, se advierte el mismo pico que en la mayoría de casos, debido a la caída de la caja al finalizar el paso.

Este caso mantiene cierta similitud con la del codo en el eje “y”, pero con valores mucho menores, debido a que las muñecas no tienen que sostener del todo la caja.

La mayoría de los valores son positivos, siguiendo la regla de la mano derecha.

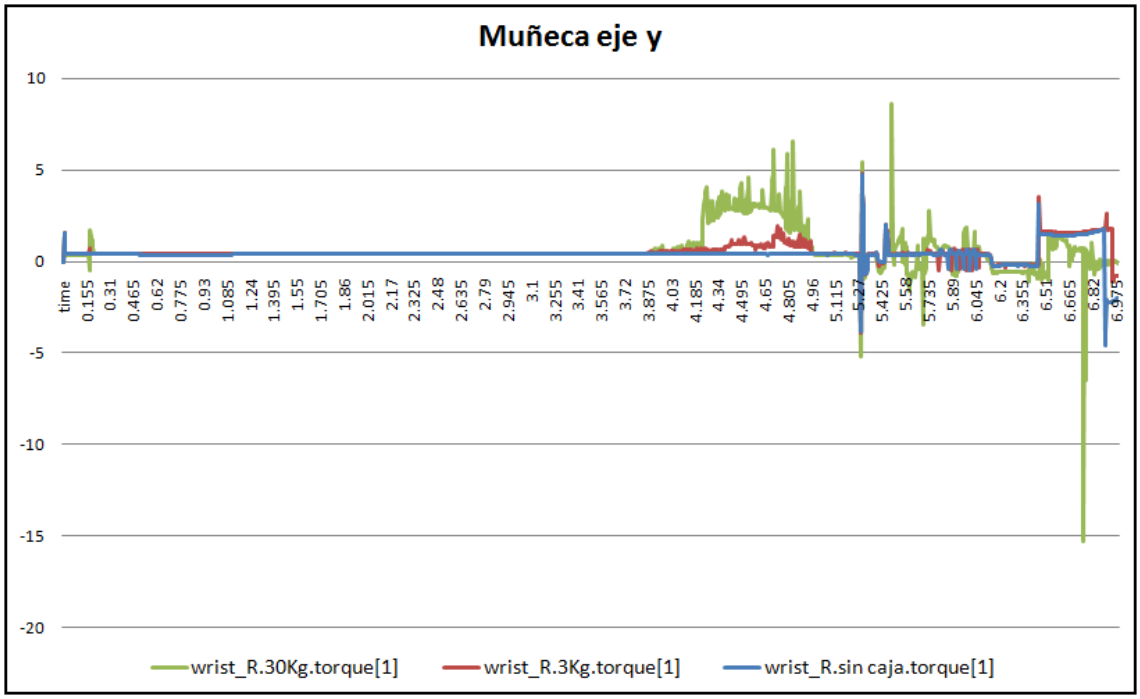


**Figura 5.66 Muñeca izquierda eje Y**

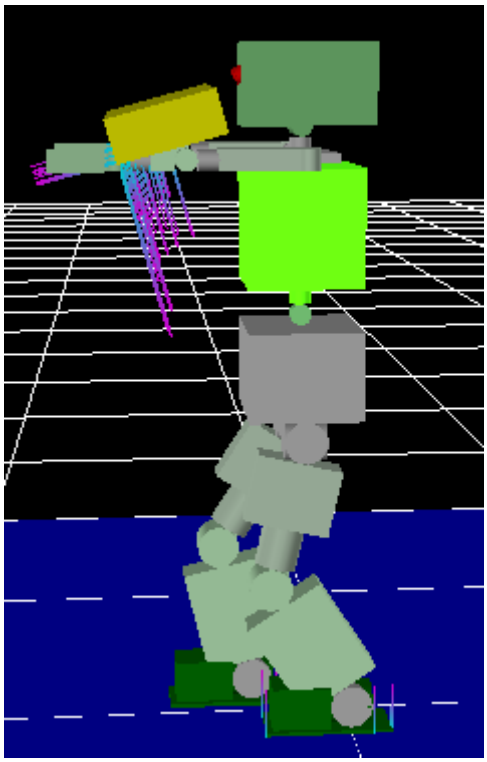

**Figura 5.67 Caja de 30 Kg se inclina**

### 4.5.3.3 Eje Z

Con respecto al eje “z”, los datos también son pequeños. El hecho de que los pares de la caja de 3 Kg y sin caja sean parecidos (ver figura 5.42), se debe a que estamos hablando de fuerzas laterales, inapreciables.

Tras finalizar el paso, se advierte un pico positivo en el caso de la caja de 30 Kg, debido a la caída de la caja, y uno negativo en los otros dos, en los que el robot se mantiene en pie.

Cabe destacar que en el caso del robot sin caja, hay un incremento de valores desde que termina el primer paso hasta que eleva el pie izquierdo. A continuación, aparece un rango de valores negativos, mientras está dando el paso con el pie izquierdo, causado por la inclinación hacia la derecha del robot. Posteriormente, se aprecia un pico al posar el pie izquierdo, y otro pico al finalizar la simulación (no se debe tener en cuenta ya que al finalizar la simulación el robot se queda en balanceo).

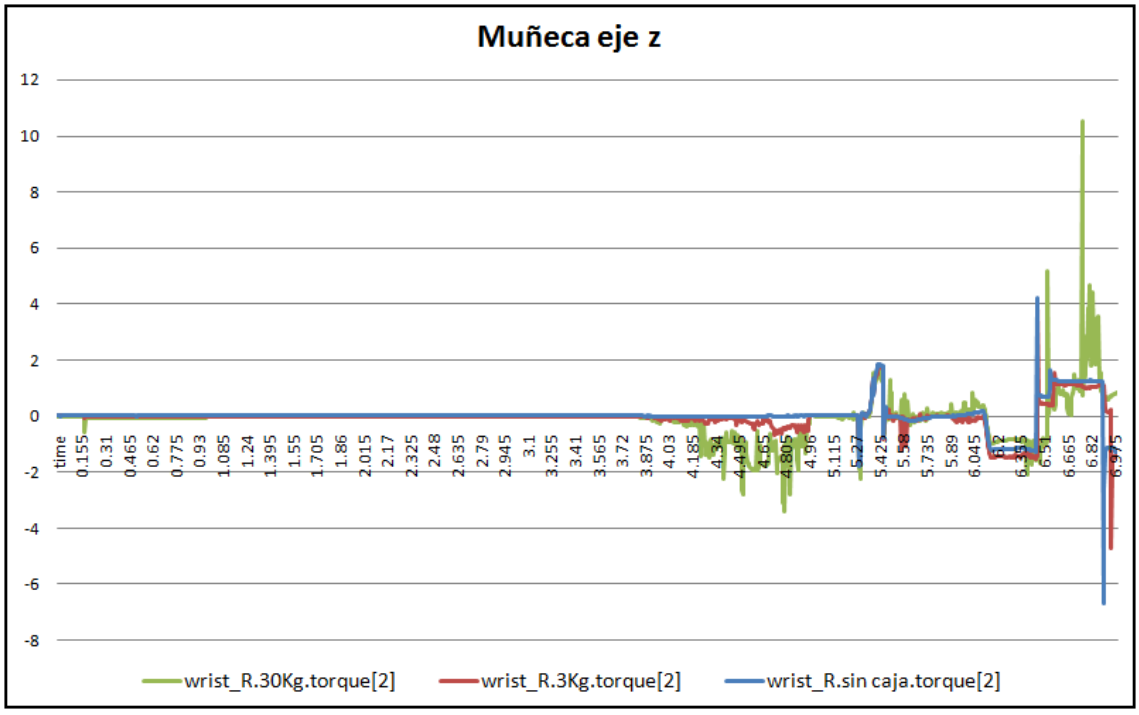


**Figura 5.68 Muñeca izquierda eje Z**

## 4.5.4 Tobillo izquierdo y derecho

Se observa una diferencia considerable entre ambos tobillos en cuanto a pares de fuerza se refiere. Esto es debido a la inexactitud del simulador, puesto que en el momento en el que el robot está flexionando las piernas, los pares deberían ser si no idénticos, parecidos.

### 4.5.4.1 Eje X

#### 4.5.4.1.1 *Tobillo izquierdo*

Como se muestra en la figura 5.43, los valores de sin caja, con caja de 3 Kg y con caja de 30 Kg, son semejantes, debido a que, si en el codo y la muñeca las fuerzas laterales se incrementaban mínimamente por la aplicación del peso de la caja, en el caso del tobillo la resultante es mucho menor.

Se puede distinguir un valor negativo continuo, por el peso interno del robot (regla de la mano derecha al contrario), mientras el robot flexiona las rodillas y se inclina (al inclinarse disminuye puesto que lo hace hacia ese lado).

Al empezar el paso con el pie derecho, ofrece un rango de valores positivos, donde el robot está apoyado sobre ese tobillo. Posteriormente, se divisa un pico negativo que aparece al apoyar el pie derecho (ya explicado con anterioridad). A continuación, se contempla un rango de valor cero, en el que el tobillo izquierdo está en el aire dando el segundo paso.

Tras el apoyo de este tobillo, aparece otro máximo negativo y tras esto dos casos más: para el caso de sin caja y con caja de 3 Kg, el robot hace un sobreesfuerzo en el pie izquierdo causado por su inclinación en el balanceo; y para el caso de 30 Kg, en el que el robot cae, hay un pico y unos valores negativos debido a la inestabilidad de la caída.

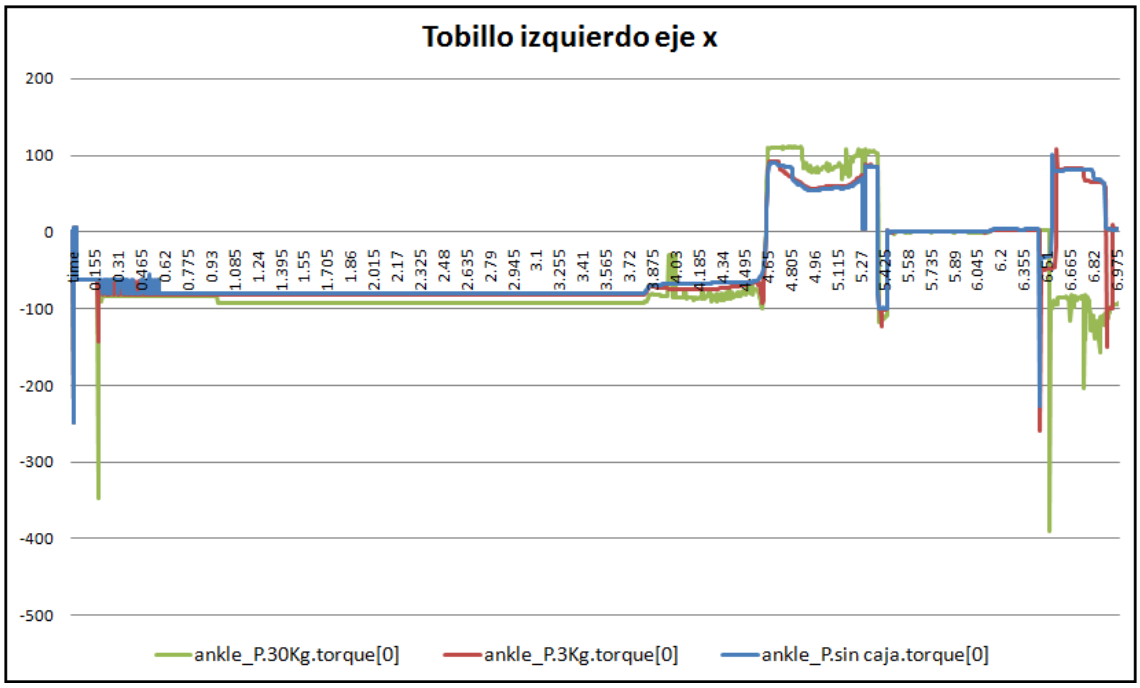


**Figura 5.69 Tobillo izquierdo eje X**

#### 4.5.4.1.2 *Tobillo derecho*

Se aprecia en la figura 5.44 una semejanza al principio con la anterior gráfica del tobillo izquierdo ya que hasta que empieza a dar el paso, los dos pies permanecen apoyados.

Cuando se eleva el pie derecho para comenzar a dar el paso, se aprecia que el par pasa a valer cero, habiendo un pico de 200 N*m en el momento de posar el pie.

Después, donde los pares del tobillo izquierdo valían cero, aquí no, puesto que el robot se sujeta sobre el pie derecho al dar el segundo paso. Estos valores son negativos, ya que el robot está inclinado hacia la derecha, y según la regla de la mano derecha invertida, queda este sentido.

Luego pasa a valer cero al terminar el paso, ya que el robot queda mayormente apoyado sobre la pierna izquierda, habiendo un pico negativo debido al balanceo (del pie izquierdo pasa al pie derecho). Se distingue cómo dicho pico llega más tarde con la caja de 3 Kg, ya que el robot, al tener más peso, se inclina más sobre la punta de su pie izquierdo en la finalización del paso, tardando en retornar al pie derecho (acción de tambaleo).

Esto no se aprecia para el caso de la caja de 30 Kg, ya que el robot se cae al suelo.

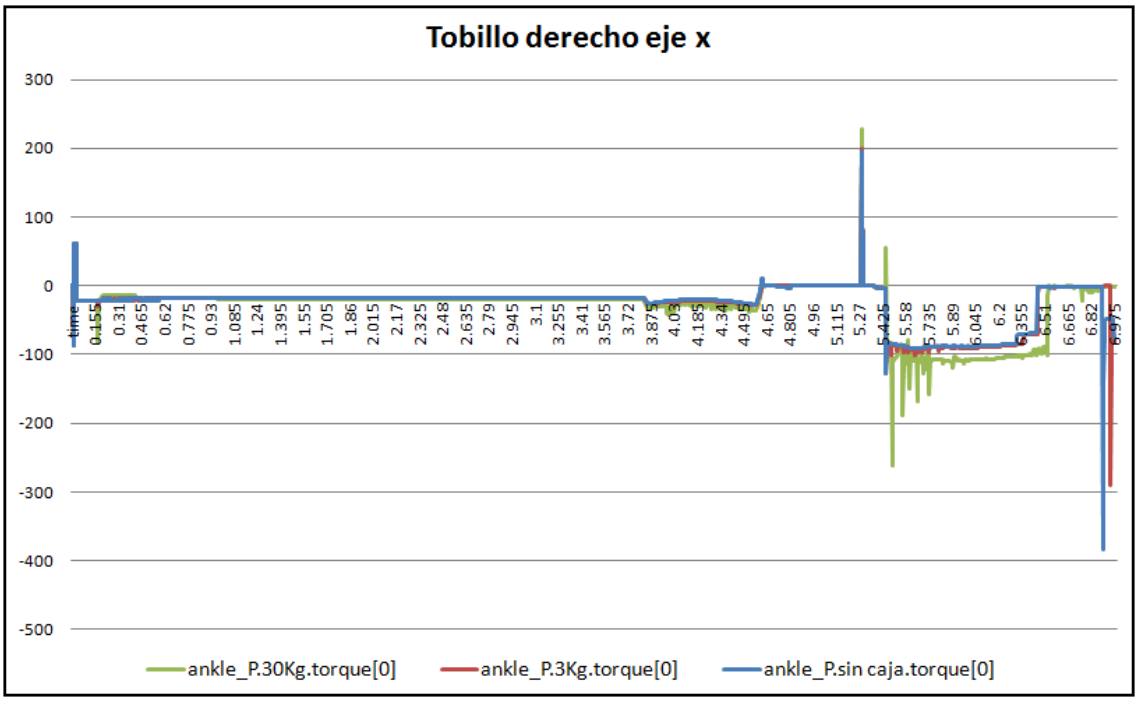


**Figura 5.70 Tobillo derecho eje X**

### 4.5.4.2 Eje Y

En el eje “y”, al igual que en los brazos, es donde mayores pares existen.

#### 4.5.4.2.1 *Tobillo izquierdo*

Como en la mayoría de casos, y con más razón aún, en el caso del eje “y”, se observa (ver figura 5.45) al principio de la gráfica el pico de valor negativo al caer la caja sobre los brazos. Éste alcanza un valor de 1700 N*m.

Más adelante, se observa un valor continuo negativo, el cual se incrementa ligeramente al dar el primer paso con el pie derecho. Al depositar éste, se genera un pico de aproximadamente 100 N*m, que seguidamente se convierte en otro de valor negativo de 300 N*m al elevar el pie izquierdo para dar el segundo paso. En esta fase pasa a valer cero, debido a que el pie está en el aire, hasta finalizar el paso, donde se divisa un máximo negativo de valor 1050 N*m para el caso de la caja de 30 Kg.

Después, en la fase de tambaleo, para el caso de sin caja y de caja de 3 Kg se observa un valor constante, no así para el ejemplo de 30 Kg, donde se nota la distorsión debida a la caída de la caja.

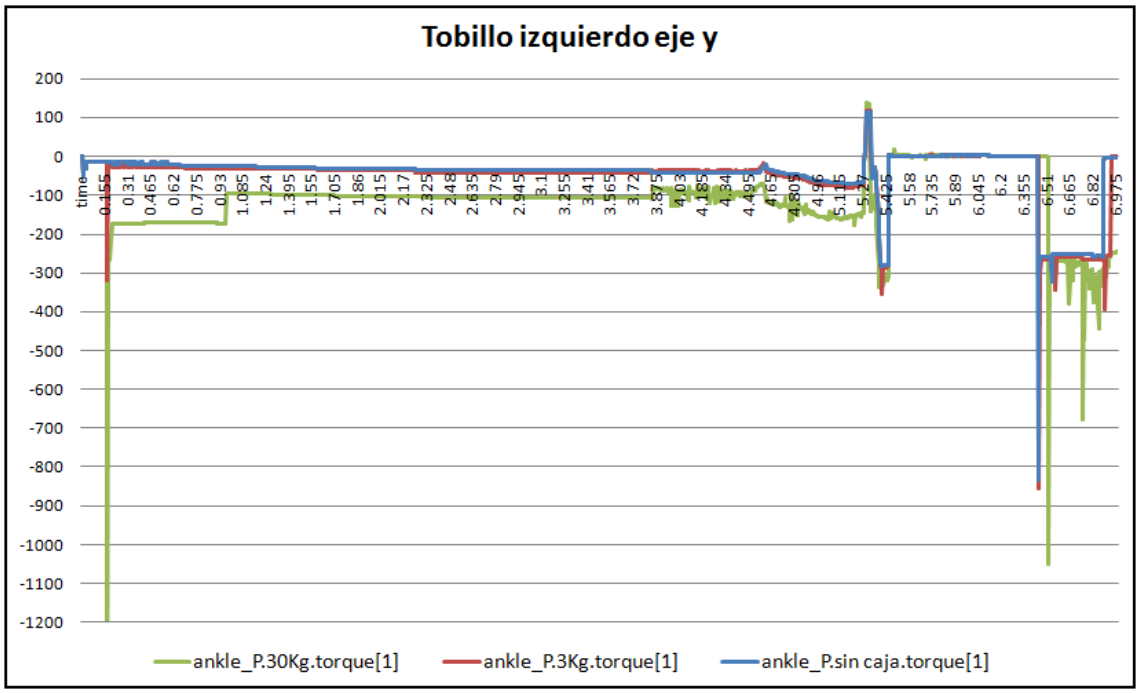


**Figura 5.71 Tobillo izquierdo eje Y**

#### 4.5.4.2.2 *Tobillo derecho*

Inicialmente, el tobillo derecho tiene cierta similitud gráficamente con el izquierdo, debido a que el robot está recto (Ver figura 5.46).

En el segundo 4.860, al comenzar el paso derecho, se observa que el valor es cero, hasta el momento de la pisada.

Posteriormente, al elevar el pie izquierdo para dar el segundo paso, y quedar el robot sostenido en el pie derecho, destaca una zona turbulenta para el caso de los 30 Kg, debido a la inclinación y rotación de la caja. Algo menos, pero también turbulenta, sucede para el caso de 3 Kg y sin caja, donde el paso izquierdo parece generar alteraciones en la coherencia de los datos.

En el segundo 6.585, al finalizar el paso completo, se distingue cómo el valor pasa a ser cero, a causa del comienzo del balanceo sobre el pie izquierdo (pie derecho en el aire). El retorno del balanceo, donde el robot se vence sobre el pie derecho, se muestra en la gráfica con un pico de valor negativo al final. Este pico se aprecia ligeramente más tarde en el caso de la caja de 3 Kg que en el ejemplo sin caja, debido a lo explicado anteriormente. Esto no es apreciable para el caso de la caja de 30 Kg, ya que el robot se precipita.

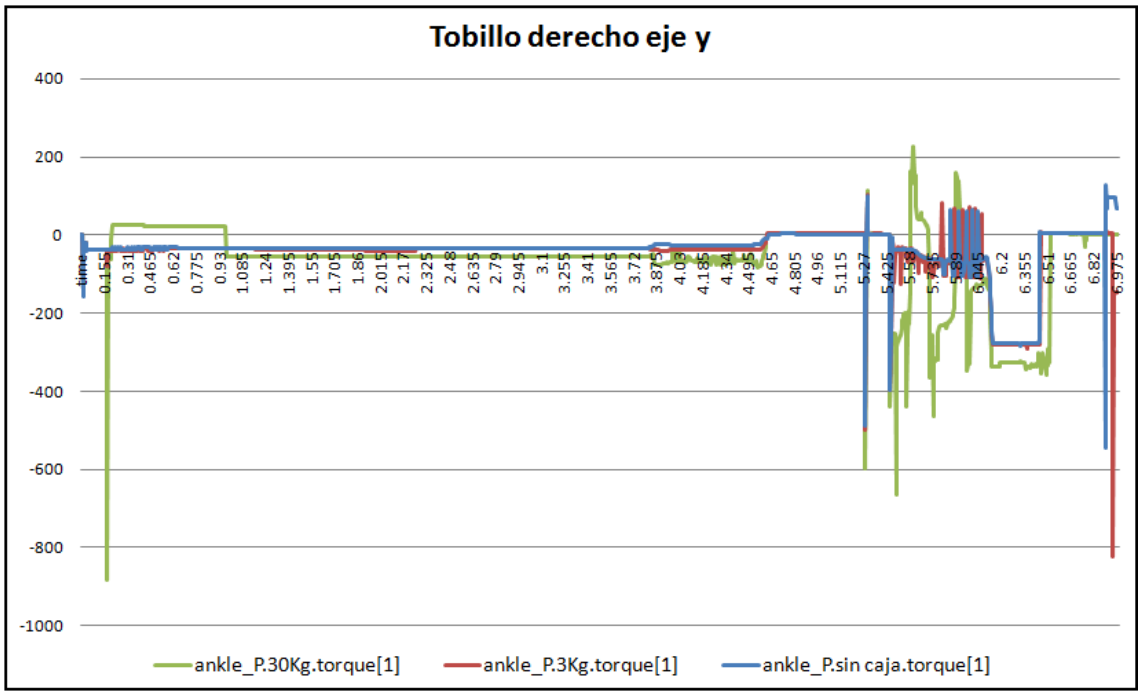


**Figura 5.72 Tobillo derecho eje Y**

### 4.5.4.3 Eje Z

Este eje es el que menos pares siente, debido a que se alimenta de fuerzas laterales, y la caja pesa con sentido vertical, y por tanto dichas fuerzas sólo pueden aparecer al inclinarse el robot.

Los datos para los casos de sin caja y con caja de 3 Kg son prácticamente iguales, ya que las pequeñas fuerzas laterales presentes, no varían de manera perceptible con un incremento vertical de 3 Kg.

#### 4.5.4.3.1 *Tobillo izquierdo*

Como es lógico, y a pesar de los picos al inicio de la simulación (la caída de la caja puede provocar alguna distorsión lateral), los valores para los tres casos valen cero hasta alcanzar el momento de la inclinación para dar el paso (segundo 3.910).

Se observa con claridad en la figura 5.47 el momento del paso izquierdo, con valores de pares negativos, y el del paso derecho, donde los valores son cero prácticamente.

Finalmente, para los casos de sin caja y con 3 Kg, se aprecia el pico negativo del momento de pisar al finalizar el paso con el pie izquierdo, manteniéndose en un valor de aproximadamente 20 N*m negativos, hasta que el tambaleo lo convierte en cero, posándose el robot sobre su pie derecho.

Lo ocurrido al final con respecto al caso de la caja de 30 Kg, es la irregularidad y pico positivo debido a la caída del robot.

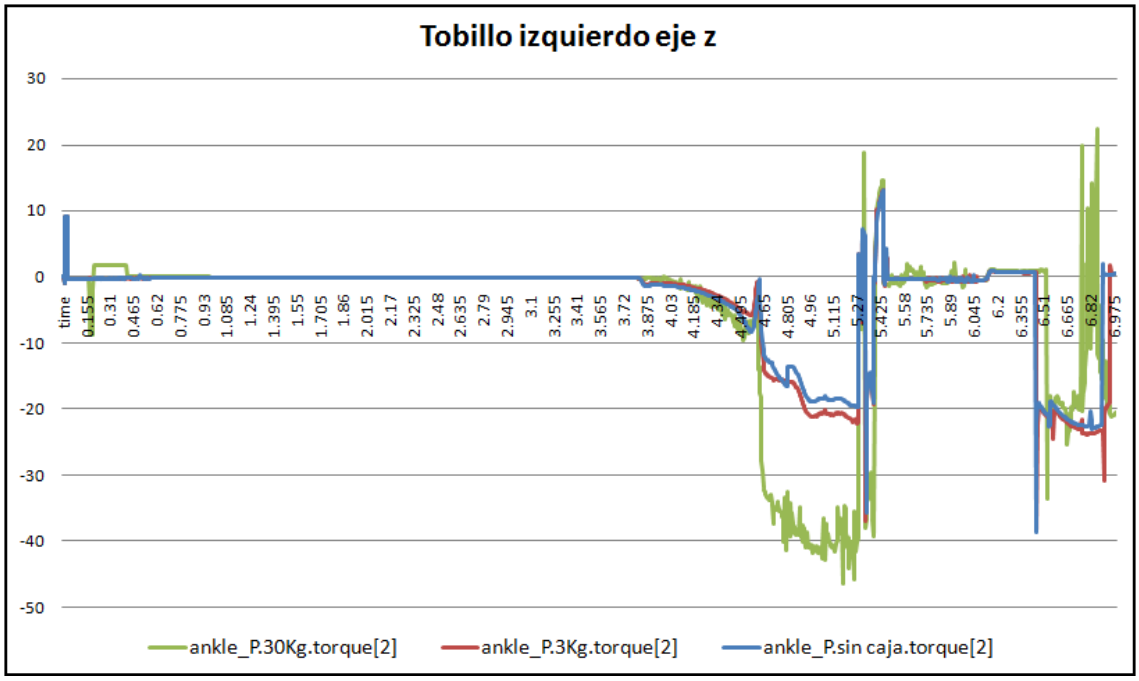


**Figura 5.73 Tobillo izquierdo eje Z**

#### 4.5.4.3.2 *Tobillo derecho*

En la figura 5.48 se observa que los valores de los pares causados por fuerzas laterales para el tobillo derecho son mayores que para el izquierdo.

Al igual que en el eje “y”, se aprecian turbulencias mientras el robot está ejecutando el segundo paso con su pie izquierdo, dando valores altos debido a la rotación de la caja en ese segundo paso.

En el intervalo del primer paso y la finalización del segundo (robot se sostiene sobre la punta de su pie izquierdo) se aprecia como los valores son cero, estando el pie derecho en el aire.

Finalmente, para los casos en los que el robot no se cae, hay un pico debido a la vuelta del tambaleo sobre su pie derecho. También se distingue el desfase en este último pico, causado por el peso de la caja, que tira hacia delante al robot, durando así más tiempo el balanceo.

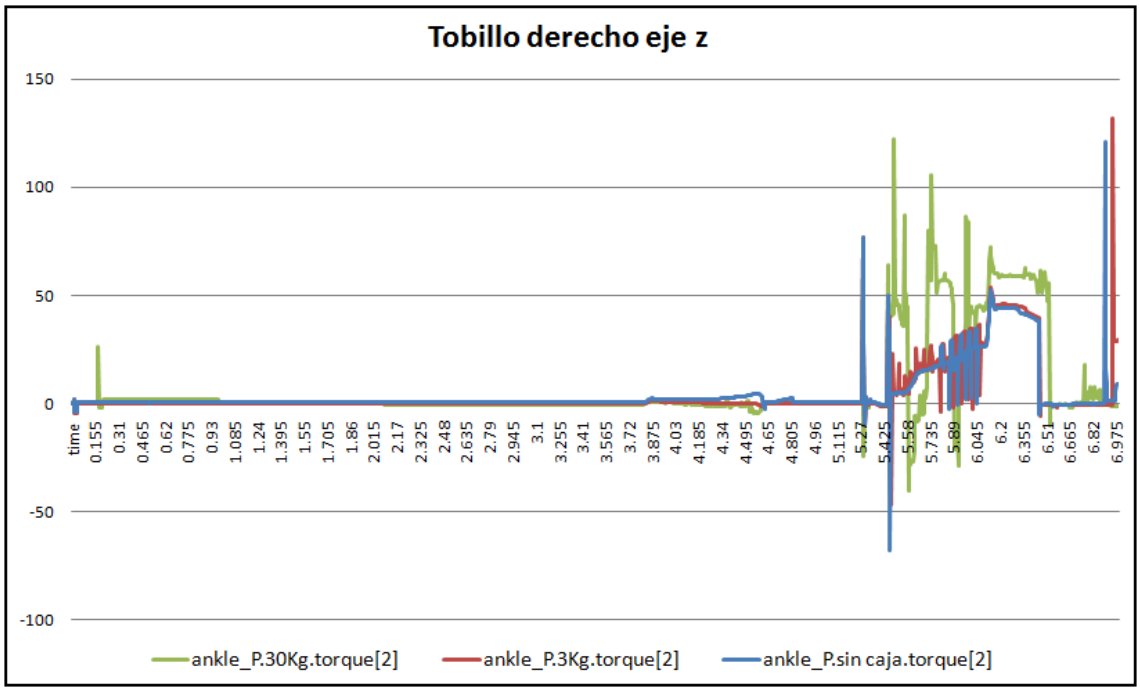


**Figura 5.74 Tobillo derecho eje Z**

# 5 Conclusiones

Una vez entendido el funcionamiento y la dinámica del simulador, se ha podido poner en práctica la visualización del robot Rh-2 con la caja sostenida en los brazos.

Han sido comprobadas y verificadas las variaciones en los pares de fuerza de las articulaciones con las variaciones en el peso de la caja sostenida por el robot.

Con esto, y como es lógico, se concluye que a mayor peso sostenido, mayor fuerza sobre sus brazos.

No obstante, los resultados obtenidos al finalizar la simulación con la caja de 30 Kg, no son fiables, dado el error mostrado de la caída de la caja a través de los brazos.

Por esta razón, el proyecto tiene como fundamento demostrar las variaciones con el peso de la caja, y no las fuerzas que siente el robot sobre los brazos. Esto es una manera de comprobar la veracidad y coherencia del simulador. Y queda demostrado.

# 6 Trabajos futuros

Como desarrollos futuros pueden incluirse las siguientes líneas de estudio:

- Desarrollar una interfaz de datos que permita adaptar las tareas creadas con otros programas a las condiciones específicas del simulador OpenHRP3. Es decir, una interfaz que permita una realización más sencilla de los ficheros ".dat" que contienen las posiciones y velocidades para las tareas.

- Estudio en profundidad del programa dentro del simulador encargado de la estabilidad del modelo durante la simulación.

- Mejorar el fichero VRML tanto del modelo del robot como de su entorno. Se puede perfeccionar la apariencia física del robot para que sea lo más semejante al robot RH-2, y no limitarla a cilindros y prismas.

- Mejorar y evitar el error del simulador en el que se fusionan el robot y un elemento externo.

# 7 Bibliografía

MANUALES

[1] “OpenHRP3 Course”. General Robotix Inc, 2008.

LIBROS

[2] “3D modeling and animation: synthesis and analysis techniques for the human body”, Nikos Sarris,Michael G. Strintzis.

TESIS DE MÁSTER

[3] “Plataforma de simulación cinemática y dinámica de futuras versiones del robot Asibot”, Tesis de Máster de Carlos Pérez de la Fuente. Noviembre 2008.

PROYECTOS FIN DE CARRERA

[4] “Simulación de la plataforma robótica HOAP-3 en el simulador OpenHRP3”, Proyecto Fin de Carrera de Tamara Ramos Cambero. Septiembre 2009.

[5] “Modelado del robot humanoide Rh-2 en la plataforma de simulación OpenHRP”, Proyecto Fin de Carrera de Rubén Manuel Sierra Molina. Julio 2010.

DIRECCIONES DE INTERNET

[6] http://www.web3d.org/

[7] http://www.uc3m.es/

[8] http://www.openrtp.jp

[9] http://www.is.aist.go.jp

[10] http://www.microsoft.com

[11] http://www.boost.org

[12] http://www.netlib.org/clapack

[13] http://tvmet.sourceforge.net

[14] http://www.wikipedia.com

[15] http://omniorb.sourceforge.net

[16] http://www.python.org

[17] http://www.java.com

[18] http://www.jython.org

# 8 Anexos

# 8.1 Sample.wrl

```
#VRML V2.0 utf8
#----------------------------------------------------------------
# OpenHRP Sample Model
#
# author    Ichitaro Kohara (YNL, Univ. of Tokyo)
# version   1.0 (2000.11.08)
# modified  Hirohisa Hirukawa (ETL)
# version   1.1 (2000.11.24)
# modified  Natsuki Miyata (MEL)
# version   1.1 (2000.12.7)
#----------------------------------------------------------------

PROTO Joint [
  exposedField     SFVec3f      center              0 0 0
  exposedField     MFNode       children            []
  exposedField     MFFloat      llimit              []
  exposedField     MFFloat      lvlimit             []
  exposedField     SFRotation   limitOrientation    0 0 1 0
  exposedField     SFString     name                ""
  exposedField     SFRotation   rotation            0 0 1 0
  exposedField     SFVec3f      scale               1 1 1
  exposedField     SFRotation   scaleOrientation    0 0 1 0
  exposedField     MFFloat      stiffness           [ 0 0 0 ]
  exposedField     SFVec3f      translation         0 0 0
  exposedField     MFFloat      ulimit              []
  exposedField     MFFloat      uvlimit             []
  exposedField     SFString     jointType           ""
  exposedField     SFInt32      jointId             -1
  exposedField     SFString     jointAxis           "Z"

  exposedField     SFFloat      gearRatio           1
  exposedField     SFFloat      rotorInertia        0
  exposedField     SFFloat      rotorResistor       0
  exposedField     SFFloat      torqueConst         1
  exposedField     SFFloat      encoderPulse        1
]
{
  Transform {
    center           IS center
    children         IS children
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
  }
}

PROTO Segment [
  field           SFVec3f     bboxCenter        0 0 0
  field           SFVec3f     bboxSize          -1 -1 -1
  exposedField    SFVec3f     centerOfMass      0 0 0
  exposedField    MFNode      children          [ ]
  exposedField    SFNode      coord             NULL
  exposedField    MFNode      displacers        [ ]
  exposedField    SFFloat     mass              0
  exposedField    MFFloat     momentsOfInertia  [ 0 0 0 0 0 0 0 0 0 ]
  exposedField    SFString    name              ""
  eventIn         MFNode      addChildren
```

```
  eventIn         MFNode      removeChildren
]
{
  Group {
    addChildren    IS addChildren
    bboxCenter     IS bboxCenter
    bboxSize       IS bboxSize
    children       IS children
    removeChildren IS removeChildren
  }
}

PROTO Humanoid [
  field           SFVec3f    bboxCenter            0 0 0
  field           SFVec3f    bboxSize              -1 -1 -1
  exposedField    SFVec3f    center                0 0 0
  exposedField    MFNode     humanoidBody          [ ]
  exposedField    MFString   info                  [ ]
  exposedField    MFNode     joints                [ ]
  exposedField    SFString   name                  ""
  exposedField    SFRotation rotation              0 0 1 0
  exposedField    SFVec3f    scale                 1 1 1
  exposedField    SFRotation scaleOrientation      0 0 1 0
  exposedField    MFNode     segments              [ ]
  exposedField    MFNode     sites                 [ ]
  exposedField    SFVec3f    translation           0 0 0
  exposedField    SFString   version               "1.1"
  exposedField    MFNode     viewpoints            [ ]
]
{
  Transform {
    bboxCenter       IS bboxCenter
    bboxSize         IS bboxSize
    center           IS center
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
    children [
      Group {
        children IS viewpoints
      }
      Group {
        children IS humanoidBody
      }
    ]
  }
}

PROTO VisionSensor [
  exposedField SFVec3f    translation       0 0 0
  exposedField SFRotation rotation          0 0 1 0
  exposedField MFNode     children          [ ]
  exposedField SFFloat    fieldOfView       0.785398
  exposedField SFString   name              ""
  exposedField SFFloat    frontClipDistance 0.01
  exposedField SFFloat    backClipDistance  10.0
  exposedField SFString   type              "NONE"
  exposedField SFInt32    sensorId          -1
  exposedField SFInt32    width             320
  exposedField SFInt32    height            240
```

```
  exposedField SFFloat    frameRate         30
]
{
  Transform {
    rotation         IS rotation
    translation      IS translation
    children         IS children
  }
}


PROTO ForceSensor [
  exposedField SFVec3f    maxForce    -1 -1 -1
  exposedField SFVec3f    maxTorque   -1 -1 -1
  exposedField SFVec3f    translation 0 0 0
  exposedField SFRotation rotation    0 0 1 0
  exposedField SFInt32    sensorId    -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO Gyro [
  exposedField SFVec3f    maxAngularVelocity -1 -1 -1
  exposedField SFVec3f    translation        0 0 0
  exposedField SFRotation rotation           0 0 1 0
  exposedField SFInt32    sensorId           -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO AccelerationSensor [
  exposedField SFVec3f    maxAcceleration -1 -1 -1
  exposedField SFVec3f    translation     0 0 0
  exposedField SFRotation rotation        0 0 1 0
  exposedField SFInt32    sensorId        -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO PressureSensor [
  exposedField SFFloat    maxPressure -1
  exposedField SFVec3f    translation 0 0 0
  exposedField SFRotation rotation    0 0 1 0
  exposedField SFInt32    sensorId    -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
```

```
  }
}

PROTO PhotoInterrupter [
  exposedField SFVec3f transmitter 0 0 0
  exposedField SFVec3f receiver    0 0 0
  exposedField SFInt32 sensorId    -1
]
{
  Transform{
    children [
      Transform{
        translation IS transmitter
      }
      Transform{
        translation IS receiver
      }
    ]
  }
}

PROTO CylinderSensorZ [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 1 0 0 1.5708
    children [
      DEF SensorY CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisY Transform{
        children [
          Transform{
            rotation 1 0 0 -1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorY.rotation_changed TO AxisY.set_rotation
}

PROTO CylinderSensorY [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 0 1 0 1.5708
    children [
      DEF SensorX CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisX Transform{
```

```
        children [
          Transform{
            rotation 0 1 0 -1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorX.rotation_changed TO AxisX.set_rotation
}

PROTO CylinderSensorX [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 0 0 1 -1.5708
    children [
      DEF SensorZ CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisZ Transform{
        children [
          Transform{
            rotation 0 0 1 1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorZ.rotation_changed TO AxisZ.set_rotation
}

NavigationInfo {
  avatarSize    0.5
  headlight     TRUE
  type  ["EXAMINE", "ANY"]
}

Background {
  skyColor 0.4 0.6 0.4
}

Viewpoint {
  position    3 0 0.835
  orientation 0.5770 0.5775 0.5775 2.0935
}

DEF SampleRobot Humanoid {
  name "sample"
  version "1.1"
  info [
    "This is a sample model of OpenHRP."
    "You can modify and use this model freely."
    "Author  : Ichitaro Kohara, YNL, Univ. of Tokyo"
    "Date    : 2000.11.08"
```

```
  "Modifying Author  : Natsuki Miyata, MEL"
  "Date    : 2000.12.08"
  "Version : 1.1"
]

humanoidBody [

  DEF WAIST Joint {
    jointType "free"
    translation 0 0 0.76975
    children [
    DEF sensor0 ForceSensor { sensorId 0 }
      DEF gsensor AccelerationSensor { sensorId 0 }
      DEF gyrometer Gyro { sensorId 0 }
      DEF WAIST_LINK0 Segment {
        centerOfMass  0 0 0.0375
        mass    27.0
        momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
        children [
          Shape {
            appearance DEF WAIST_APP Appearance {
              material Material {
              }
            }
            geometry Box { size 0.08 0.11754 0.08 }
          }
          Transform {
            translation   0 0 0.144
            children      Shape {
              appearance  USE WAIST_APP
              geometry Box { size 0.268 0.34 0.208 }
            }
          }
        ]
      }

      DEF WAIST_P Joint {


      jointType "rotate"
        jointAxis "Y"

        jointId  24
         translation     0 0 0.273
        children [
        DEF sensor1 ForceSensor { sensorId 1 }
          DEF WAIST_LINK1 Segment {
            centerOfMass  0 0 -0.1
            mass    6.0
            momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
            children [
              Shape {
                appearance  DEF WAIST_LINK1_APP Appearance {
                  material  Material {
                    diffuseColor   0.6 1.0 0.6
                  }
                }
                geometry Cylinder { radius 0.025 height 0.05 }
              }
            ]
          }
```

```
              DEF WAIST_Y Joint {
                jointType "rotate"
              jointAxis "Z"
                jointId  25
                children [
                DEF sensor2 ForceSensor { sensorId 2 }
                  DEF WAIST_LINK2 Segment {
                    centerOfMass  0.11 0 0.25
                    mass    30.0
                    momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
                    children [
                      Transform {
                     rotation 1 0 0 1.570
                     translation 0 0 0.05
                        children Shape {
                          appearance DEF WAIST_LINK2_APP Appearance {
                            material      Material {
                              diffuseColor       0.6 1.8 0.1
                            }
                          }
                          geometry Cylinder { radius 0.025 height 0.05 }
                        }
                      }
                      Transform {
                        translation 0 0 0.2125
                        children  Shape {
                          appearance USE WAIST_LINK2_APP
                          geometry Box { size 0.268 0.34 0.275 }
                        }
                      } # Transform
                    ]
                  } # segment WAIST_LINK2

                  DEF CHEST_Y Joint {
                    jointType "rotate"
                    jointId  26
                    translation 0 0 0.383015
                    children [
                    DEF sensor3 ForceSensor { sensorId 3 }
                      DEF WAIST_LINK3 Segment {
                        centerOfMass  0 0 0
                        mass    13.0
                        momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
                        children [
                          Transform {
                            rotation 1 0 0 1.5708
                            children Shape {
                              appearance DEF WAIST_LINK3_APP Appearance
{
                                material  Material {
                                  diffuseColor 0.8 0.8 0.8
                                }
                              }
                              geometry Cylinder { radius 0.025 height
0.06603 }
                            }
                          }
                          Transform {
                        translation 0 0 -0.01
                        children  Shape {
                         appearance USE WAIST_LINK3_APP
```

```
                    geometry Box { size 0.15 0.34 0.02 }
                  }

                } # Transform



                  ]
                  }

                DEF CHEST_P Joint {
              jointType "rotate"
              jointAxis "Y"
              jointId  27
              translation 0 0 -0.006
              children [
              DEF sensor4 ForceSensor { sensorId 4 }
                DEF WAIST_LINK4 Segment {
                  centerOfMass  0 0 0
                  mass    13.0
                  momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
                  children [
                    Transform {

                      translation 0 0 0.065
                      children  Shape {
                        appearance      Appearance {
                          material      Material {
                            diffuseColor  0.5 0.8 0.5
                          }
                        }
                        geometry Cylinder { radius 0.025 height
0.1 }
                      }
                    }
                    Transform {
                      translation -0.015 0 0.16
                      children  Shape {
                        appearance      Appearance {
                          material      Material {
                            diffuseColor 0.5 0.8 0.5
                          }
                        }
                        geometry Box { size 0.31 0.19 0.19 }
                      } # shape
                    } # transform
                  ]
                } # segment WAIST_LINK4

                DEF VISION_SENSOR1 VisionSensor {
                  translation   0.138 0.05 0.18
                  rotation      0.4472 -0.4472 -0.7746 1.8235
                  name          "LeftCamera"
                  type          "DEPTH"
                  sensorId      0
                  children [
                  DEF sensor5 ForceSensor { sensorId 5 }
                    DEF CAMERA_SHAPE Transform {
                      rotation  1 0 0 -1.5708
                      children [
                        Shape {
```

```
                geometry Cylinder {
                  radius      0.02
                  height      0.025
                }
                appearance Appearance {
                  material Material {
                    diffuseColor      1 0 0
                  }
                }
              }
            ]
          }
        ]
      }
      DEF VISION_SENSOR2 VisionSensor {
        translation   0.138 -0.05 0.18
        rotation      0.4472 -0.4472 -0.7746 1.8235
        name          "RightCamera"
        type          "DEPTH"
        sensorId      1
        children [
          USE CAMERA_SHAPE
        ]
      }

     #==================== Left Arm ====================

      DEF LARM_SHOULDER_P Joint {
        jointType "rotate"
      jointAxis "Y"
        jointId  18
        translation   0 0.215 -0.005
        children [
        DEF sensor6 ForceSensor { sensorId 6 }
          DEF LARM_LINK1 Segment {
            centerOfMass  0.1 0 0
            mass    3.0
            momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
            children Transform {
              translation 0 -0.035 0
              children DEF ARM_SHAPE1 Shape {
                appearance Appearance {
                  material Material {
                  }
                }
                geometry Cylinder { radius 0.025 height
0.02 }
              }
            }
          }

          DEF LARM_SHOULDER_R Joint {
            jointType "rotate"
      jointAxis "X"
            jointId  19
            children [
            DEF sensor7 ForceSensor { sensorId 7 }
              DEF LARM_LINK2 Segment {
                centerOfMass  0 0 -0.1
                mass    0.6
                momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
```

```
                            children DEF ARM_SHAPE2 Transform {
                              children [
                                Transform {
                                  rotation 0 0 1 1.5708
                                  children Shape {
                                    appearance DEF ARM_LINK2_APP
Appearance {
                                      material  Material {
                                        diffuseColor 0.8 0.9 0.8
                                      }
                                    }
                                    geometry Cylinder { radius 0.025
height 0.05 }
                                  }
                                }
                                Transform {
                                  translation   0 0 -0.1140875
                                  children      Shape {
                                    appearance USE ARM_LINK2_APP
                                    geometry Box { size 0.05 0.05
0.178175 }
                                  }
                                }
                              ]
                            }
                          } # Segment LARM_LINK2

                          DEF LARM_SHOULDER_Y Joint {
                            jointType "rotate"
                            jointId  20
                            translation     0 0 -0.271825
                            children [
                            DEF sensor8 ForceSensor { sensorId 8 }
                              DEF LARM_LINK3 Segment {
                                centerOfMass  0 0 0
                                mass    1.0
                                momentsOfInertia [ 1 0 0 0 1 0 0 0 1
]
                                children DEF ARM_SHAPE3 Transform {
                                  translation 0 0 0.05
                                  rotation 1 0 0 1.5708
                                  children Shape {
                                    appearance Appearance {
                                      material Material {
                                      }
                                    }
                                    geometry Cylinder { radius 0.025
height 0.05}
                                  }
                                }
                              } # Segment LARM_LINK3

                              DEF LARM_ELBOW Joint {
                                jointType "rotate"
                        jointAxis "Y"
                                jointId  21
                                children [
                                DEF sensor9 ForceSensor { sensorId 9
}
                                  DEF LARM_LINK4 Segment {
                                    centerOfMass  0 0 -0.3
```

```
                                    mass    0.6
                                    momentsOfInertia [ 1 0 0 0 1 0 0
0 1 ]
                                    children DEF ARM_SHAPE4
Transform {
                                      children [
                                        Shape {
                                          appearance DEF ARM_APP4
Appearance {
                                            material    Material {
                                              diffuseColor     0.8
1.0 0.8
                                            }
                                          }
                                          geometry Cylinder { radius
0.025 height 0.05 }
                                        }
                                        Transform {
                                          translation  0 0
-0.05079375
                                          children        Shape {
                                            appearance USE ARM_APP4
                                            geometry Box { size 0.05
0.05 0.0515875 }
                                          }
                                        }
                                      ]
                                    }
                                  } # Segment LARM_LINK4

                                  DEF LARM_WRIST_Y Joint {
                                    jointType "rotate"
                              jointAxis "Z"
                                    jointId  22
                                    translation       0 0 -0.1515875
                                    children [
                                    DEF sensor10 ForceSensor
{ sensorId 10 }
                                      DEF LARM_LINK5 Segment {
                                        centerOfMass  0 0 0.1
                                        mass    0.4
                                        momentsOfInertia [ 1 0 0 0 1
0 0 0 1 ]
                                        children DEF ARM_SHAPE5
Transform {
                                          translation 0 0 0.05
                                          rotation 1 0 0 1.5708
                                          children Shape {
                                            appearance Appearance {
                                              material Material {
                                              }
                                            }
                                            geometry Cylinder
{ radius 0.025 height 0.05 }
                                          }
                                        }
                                      }


                                          DEF LARM_WRIST_R Joint {
```

```
                                        jointType "rotate"
                                  jointAxis "X"
                                        jointId  23
                                        children [
                                        DEF sensor11 ForceSensor
{ sensorId 11 }
                                          DEF LARM_LINK7 Segment
{
                                            centerOfMass  0 0
-0.1
                                            mass    0.4
                                            momentsOfInertia [ 1
0 0 0 1 0 0 0 1 ]
                                            children DEF
ARM_SHAPE7 Transform {
                                              children [
                                                Transform {
                                                  rotation 0 0 1
1.5708
                                                  children Shape
{
                                                    appearance
DEF ARM_APP7 Appearance {
                                                      material
Material {
                                                        diffuseC
olor 0.7 0.9 0.7
                                                  }
                                                }
                                                geometry
Cylinder { radius 0.025 height 0.05 }
                                              }
                                            }
                                            Transform {
                                              translation 0
0 -0.08829375
                                              children
Shape {
                                                appearance
USE ARM_APP7
                                                geometry Box
{ size 0.05 0.05 0.1265875 }
                                              }
                                            }# Transform
                                          ]
                                        }
                                      } # Segment LARM_LINK7


                                    ]
                                  } # Joint LARM_WRIST_R

                                ]
                              } # Joint LARM_WRIST_Y
                            ]
                          } # Joint LARM_ELBOW
                        ]
                      } # Joint LARM_SHOULDER_Y
                    ]
```

```
      } # Joint LARM_SHOULDER_R
    ]
  } # Joint LARM_SHOULDER_P

#==================== Right Arm ====================

  DEF RARM_SHOULDER_P Joint {
    jointType "rotate"
jointAxis "Y"
    jointId 6
    translation   0 -0.215 -0.005
    children [
    DEF sensor12 ForceSensor { sensorId 12 }
      DEF RARM_LINK1 Segment {
        centerOfMass          0.1 0 0
        mass                  3.0
        momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
        children Transform {
          translation 0 0.035 0
          children USE ARM_SHAPE1
        }
      }

      DEF RARM_SHOULDER_R Joint {
        jointType "rotate"
  jointAxis "X"
        jointId 7
        children [
        DEF sensor13 ForceSensor { sensorId 13 }
          DEF RARM_LINK2 Segment {
            centerOfMass          0 0 -0.1
            mass                  0.6
            momentsOfInertia      [ 1 0 0 0 1 0 0 0
1 ]

            children [
              USE ARM_SHAPE2
            ]
          } # Segment RARM_LINK2

          DEF RARM_SHOULDER_Y Joint {
            jointType "rotate"
            jointId 8
            translation     0 0 -0.271825
            children [
            DEF sensor14 ForceSensor { sensorId 14 }
              DEF RARM_LINK3 Segment {
                centerOfMass          0 0 0
                mass                  1.0
                momentsOfInertia      [ 1 0 0 0 1 0
0 0 1 ]

                children USE ARM_SHAPE3
              }

              DEF RARM_ELBOW Joint {
                jointType "rotate"
            jointAxis "Y"
                jointId 9
                children [
                DEF sensor15 ForceSensor { sensorId
15 }

                  DEF RARM_LINK4 Segment {
```

```
                                    centerOfMass          0 0 -0.3
                                    mass                  0.6
                                    momentsOfInertia      [ 1 0 0 0
1 0 0 0 1 ]
                                    children USE ARM_SHAPE4
                                  } # Segment RARM_LINK4

                                  DEF RARM_WRIST_Y Joint {
                                    jointType "rotate"
                               jointAxis "Z"
                                    jointId 10
                                    translation  0 0 -0.1515875
                                    children [
                                    DEF sensor16 ForceSensor
{ sensorId 16 }
                                      DEF RARM_LINK5 Segment {
                                        centerOfMass          0 0
0.1
                                        mass                  0.4
                                        momentsOfInertia      [ 1 0
0 0 1 0 0 0 1 ]
                                        children USE ARM_SHAPE5
                                      }


                                          DEF RARM_WRIST_R Joint {
                                            jointType "rotate"
                                     jointAxis "X"
                                            jointId 11
                                            children [
                                            DEF sensor17 ForceSensor
{ sensorId 17 }
                                              DEF RARM_LINK7 Segment
{
                                                centerOfMass
0 0 -0.1
                                                mass
0.4
                                                momentsOfInertia
[ 1 0 0 0 1 0 0 0 1 ]
                                                children [
                                                  USE ARM_SHAPE7
                                                ]
                                              } # Segment RARM_LINK7
                                            ]
                                          } # Joint RARM_WRIST_R

                                    ]
                                  } # Joint RARM_WRIST_Y
                                ]
                              } # Joint RARM_ELBOW
                            ]
                          } # Joint RARM_SHOULDER_Y
                        ]
                      } # Joint RARM_SHOULDER_R
                    ]
                  } # Joint RARM_SHOULDER_P
                  ]
                  }# Joint CHEST_P
                ]
              } # Joint CHEST_Y
```

```
      ]
    } # Joint WAIST_Y
  ]
} # Joint WAIST_P

#==================== Left Leg ====================

DEF LLEG_HIP_R Joint {
  jointType "rotate"
jointAxis "X"
  jointId 12
  translation   0 0.10877 -0.01
  children [
  DEF sensor18 ForceSensor { sensorId 18 }

    DEF LLEG_LINK1 Segment {
      centerOfMass          0 0.1 0
      mass                  2.5
      momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
      children DEF LEG_SHAPE1 Transform {
        rotation 0 0 1 1.5708
        children Shape {
          appearance Appearance {
            material Material {
            }
          }
          geometry Cylinder { radius 0.05 height 0.1 }
        }
      }
    }

    DEF LLEG_HIP_P Joint {
      jointType "rotate"
    jointAxis "Y"
      jointId 13
      children    [
          DEF sensor19 ForceSensor { sensorId 19 }


           DEF LLEG_LINK2 Segment {
             centerOfMass          0 0 -0.15
             mass                  2.0
             momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
             children DEF LEG_SHAPE2 Shape {
               appearance Appearance {
                 material Material {
                 }
               }
               geometry Cylinder { radius 0.05 height 0.1 }
             }
           }

           DEF LLEG_HIP_Y Joint {
             jointType "rotate"
             jointId 14
             translation     0 0 -0.359875
             children [
                      DEF sensor20 ForceSensor { sensorId 20 }
               DEF LLEG_LINK3 Segment {
                 centerOfMass          0 0.04 0
                 mass                  5.1
```

```
                        momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                        children DEF LEG_SHAPE3 Transform {
                          children [
                            Transform {
                              translation 0 0 0.1
                              rotation 1 0 0 1.5708
                              children Shape {
                                appearance DEF LEG_APP3 Appearance {
                                  material  Material {
                                    diffuseColor   0.8 0.9 0.8
                                  }
                                }
                                geometry Cylinder { radius 0.05
height 0.1 }
                              }
                            }
                            Transform {
                              translation 0 0 0.2299375
                              children      Shape {
                                appearance USE LEG_APP3
                                geometry Box { size 0.2 0.1
0.159875 }
                              }
                            }
                          ]
                        }
                      } # Segment LLEG_LINK3

                      DEF LLEG_KNEE Joint {
                        jointType "rotate"
                jointAxis "Y"
                        jointId 15
                        children [
                      DEF sensor21 ForceSensor { sensorId 21 }
                         DEF LLEG_LINK4 Segment {
                           centerOfMass          0 0 -0.3
                           mass                  7.0
                           momentsOfInertia      [ 1 0 0 0 1 0 0 0 1
]
                           children DEF LEG_SHAPE4 Transform {
                             children [
                               Shape {
                                 appearance DEF LEG_APP4 Appearance
{
                                   material    Material {
                                     diffuseColor     0.8 1.0 0.8
                                   }
                                 }
                                 geometry Cylinder { radius 0.05
height 0.1 }
                               }
                               Transform {
                                 translation      0 0 -0.175
                                 children        Shape {
                                   appearance USE LEG_APP4
                                   geometry Box { size 0.2 0.1
0.25 }
                                 }
                               }
                             ]
                           }
```

```
                        } # Segment LLEG_LINK4

                        DEF LLEG_ANKLE_P Joint {
                          jointType "rotate"
                    jointAxis "Y"
                          jointId 16
                          translation       0 0 -0.35
                          children [
                            DEF sensor22 ForceSensor { sensorId 22 }
                            DEF LLEG_LINK5 Segment {
                              centerOfMass          -0.15 0 0
                              mass                  2.5
                              momentsOfInertia      [ 1 0 0 0 1 0 0
0 1 ]
                              children DEF LEG_SHAPE5 Shape {
                                appearance Appearance {
                                  material Material {
                                  }
                                }
                                geometry Cylinder { radius 0.05
height 0.1 }
                              }
                            }
                            DEF LLEG_ANKLE_R Joint {
                              jointType "rotate"
                      jointAxis "X"
                              jointId 17
                              children [
                             DEF sensor23 ForceSensor { sensorId 23 }
                                 DEF LLEG_LINK6 Segment {
                                   centerOfMass          0.28 0 -0.2
                                   mass                  1.9
                                   momentsOfInertia      [ 1 0 0 0 1
0 0 0 1 ]
                                   children DEF LEG_SHAPE6 Transform
{
                                     children [
                                       Transform {
                                  #translation 0 0 0
                                            rotation 0 0 1 1.5708
                                            children Shape {
                                              appearance DEF LEG_APP6
Appearance {
                                                material      Material
{
                                                  diffuseColor 0.0 0.5
0.0
                                                }
                                              }
                                              geometry Cylinder
{ radius 0.05 height 0.1 }
                                            }
                                          }
                                           Transform {
                                            translation       0.12 0 0
                                            children  Shape {
                                              appearance USE LEG_APP6
                                              geometry Box { size 0.14
0.11 0.1 }
                                            } # Shape
                                          }#Transform
```

```
                                        Transform {
                                          translation       0.055 0
-0.0549375
                                          children  Shape {
                                            appearance USE LEG_APP6
                                            geometry Box { size 0.3
0.14 0.009875 }
                                          } # Shape
                                        }#Transform
                                      ]
                                    }
                                  } # Segment LLEG_LINK6

                                ]
                              } # Joint LLEG_ANKLE_R

                            ]
                          } # Joint LLEG_ANKLE_P

                        ]
                      } # Joint LLEG_KNEE

                    ]
                  } # Joint LLEG_HIP_Y

                ]
              } # Joint LLEG_HIP_P
            ]
          } # Joint LLEG_HIP_R

          #==================== Right Leg ====================

          DEF RLEG_HIP_R Joint {
            jointType "rotate"
          jointAxis "X"
            jointId 0
            translation   0 -0.10877 -0.01
            children [
              DEF sensor24 ForceSensor { sensorId 24 }
              DEF RLEG_LINK1 Segment {
                centerOfMass          0 -0.1 0
                mass                  2.5
                momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                children USE LEG_SHAPE1
              }

              DEF RLEG_HIP_P Joint {
                jointType "rotate"
              jointAxis "Y"
                jointId 1
                children [
                    DEF sensor25 ForceSensor { sensorId 25 }
                  DEF RLEG_LINK2 Segment {
                    centerOfMass          0 0 -0.15
                    mass                  2.0
                    momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                    children USE LEG_SHAPE2
                  }
                  DEF RLEG_HIP_Y Joint {
                    jointType "rotate"
```

```
                jointId 2
                translation     0 0 -0.359875
                children [
                DEF sensor26 ForceSensor { sensorId 26 }
                  DEF RLEG_LINK3 Segment {
                    centerOfMass          0 -0.04 0
                    mass                  5.1
                    momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                    children [
                      USE LEG_SHAPE3
                    ]
                  }

                  DEF RLEG_KNEE Joint {
                    jointType "rotate"
                jointAxis "Y"
                    jointId 3
                    children [
              DEF sensor27 ForceSensor { sensorId 27 }
                      DEF RLEG_LINK4 Segment {
                        centerOfMass          0 0 -0.3
                        mass                  7.0
                        momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                        children [
                          USE LEG_SHAPE4
                        ]
                      } # Segment RLEG_LINK4

                      DEF RLEG_ANKLE_P Joint {
                        jointType "rotate"
                  jointAxis "Y"
                        jointId 4
                        translation     0 0 -0.35
                        children [
            DEF sensor28 ForceSensor { sensorId 28 }
                          DEF RLEG_LINK5 Segment {
                            centerOfMass          -0.15 0 0
                            mass                  2.5
                            momentsOfInertia      [ 1 0 0 0 1 0 0 0
1 ]
                            children USE LEG_SHAPE5
                          }

                          DEF RLEG_ANKLE_R Joint {
                            jointType "rotate"
                        jointAxis "X"
                            jointId 5
                            children [
                DEF sensor29 ForceSensor { sensorId 29 }
                              DEF RLEG_LINK6 Segment {
                                centerOfMass          0.28 0 -0.2
                                mass                  1.9
                                momentsOfInertia      [ 1 0 0 0 1 0
0 0 1 ]
                                children [
                                  USE LEG_SHAPE6
                                ]
                              } # Segment RLEG_LINK6

                            ]
                          } # Joint RLEG_ANKLE_R
```

```
                    ]
                  } # Joint RLEG_ANKLE_P

                ]
              } # Joint RLEG_KNEE

            ]
          } # Joint RLEG_HIP_Y

        ]
      } # Joint RLEG_HIP_P

    ]
  } # Joint RLEG_HIP_R
 ]
} # Joint WAIST
]


# List up all the joints' name you use
joints [
  USE WAIST,
  USE WAIST_P,
  USE WAIST_Y,
  USE CHEST_Y,
  USE CHEST_P,

  USE LARM_SHOULDER_P,
  USE LARM_SHOULDER_R,
  USE LARM_SHOULDER_Y,
  USE LARM_ELBOW,
  USE LARM_WRIST_Y,
  USE LARM_WRIST_R,

  USE RARM_SHOULDER_P,
  USE RARM_SHOULDER_R,
  USE RARM_SHOULDER_Y,
  USE RARM_ELBOW,
  USE RARM_WRIST_Y,
  USE RARM_WRIST_R,

  USE LLEG_HIP_R,
  USE LLEG_HIP_P,
  USE LLEG_HIP_Y,
  USE LLEG_KNEE,
  USE LLEG_ANKLE_P,
  USE LLEG_ANKLE_R,

  USE RLEG_HIP_R,
  USE RLEG_HIP_P,
  USE RLEG_HIP_Y,
  USE RLEG_KNEE,
  USE RLEG_ANKLE_P,
  USE RLEG_ANKLE_R
]

# List up all the segments' name you use
segments [
  USE WAIST_LINK0,
  USE WAIST_LINK1,
```

```
    USE WAIST_LINK2,
    USE WAIST_LINK3,
    USE WAIST_LINK4,

    USE LARM_LINK1,
    USE LARM_LINK2,
    USE LARM_LINK3,
    USE LARM_LINK4,
    USE LARM_LINK5,
    USE LARM_LINK7,

    USE RARM_LINK1,
    USE RARM_LINK2,
    USE RARM_LINK3,
    USE RARM_LINK4,
    USE RARM_LINK5,
    USE RARM_LINK7,

    USE LLEG_LINK1,
    USE LLEG_LINK2,
    USE LLEG_LINK3,
    USE LLEG_LINK4,
    USE LLEG_LINK5,
    USE LLEG_LINK6,

    USE RLEG_LINK1,
    USE RLEG_LINK2,
    USE RLEG_LINK3,
    USE RLEG_LINK4,
    USE RLEG_LINK5,
    USE RLEG_LINK6
  ]
}
```

## 8.2 Box3.wrl

```
#VRML V2.0 utf8

PROTO Joint [
  exposedField     SFVec3f      center             0 0 0
  exposedField     MFNode       children           []
  exposedField     MFFloat      llimit             []
  exposedField     MFFloat      lvlimit            []
  exposedField     SFRotation   limitOrientation   0 0 1 0
  exposedField     SFString     name               ""
  exposedField     SFRotation   rotation           0 0 1 0
  exposedField     SFVec3f      scale              1 1 1
  exposedField     SFRotation   scaleOrientation   0 0 1 0
  exposedField     MFFloat      stiffness          [ 0 0 0 ]
  exposedField     SFVec3f      translation        0 0 0
  exposedField     MFFloat      ulimit             []
  exposedField     MFFloat      uvlimit            []
  exposedField     SFString     jointType          ""
  exposedField     SFInt32      jointId            -1
  exposedField     SFString     jointAxis          "Z"

  exposedField     SFFloat      gearRatio          1
  exposedField     SFFloat      rotorInertia       0
  exposedField     SFFloat      rotorResistor      0
  exposedField     SFFloat      torqueConst        1
  exposedField     SFFloat      encoderPulse       1
]
{
  Transform {
    center           IS center
    children         IS children
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
  }
}

PROTO Segment [
  field           SFVec3f     bboxCenter        0 0 0
  field           SFVec3f     bboxSize          -1 -1 -1
  exposedField    SFVec3f     centerOfMass      0 0 0
  exposedField    MFNode      children          [ ]
  exposedField    SFNode      coord             NULL
  exposedField    MFNode      displacers        [ ]
  exposedField    SFFloat     mass              0
  exposedField    MFFloat     momentsOfInertia  [ 0 0 0 0 0 0 0 0 0 ]
  exposedField    SFString    name              ""
  eventIn         MFNode      addChildren
  eventIn         MFNode      removeChildren
]
{
  Group {
    addChildren    IS addChildren
    bboxCenter     IS bboxCenter
    bboxSize       IS bboxSize
    children       IS children
    removeChildren IS removeChildren
  }
```

```
}

PROTO Humanoid [
  field           SFVec3f    bboxCenter            0 0 0
  field           SFVec3f    bboxSize              -1 -1 -1
  exposedField    SFVec3f    center                0 0 0
  exposedField    MFNode     humanoidBody          [ ]
  exposedField    MFString   info                  [ ]
  exposedField    MFNode     joints                [ ]
  exposedField    SFString   name                  ""
  exposedField    SFRotation rotation              0 0 1 0
  exposedField    SFVec3f    scale                 1 1 1
  exposedField    SFRotation scaleOrientation      0 0 1 0
  exposedField    MFNode     segments              [ ]
  exposedField    MFNode     sites                 [ ]
  exposedField    SFVec3f    translation           0 0 0
  exposedField    SFString   version               "1.1"
  exposedField    MFNode     viewpoints            [ ]
]
{
  Transform {
    bboxCenter       IS bboxCenter
    bboxSize         IS bboxSize
    center           IS center
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
    children [
      Group {
        children IS viewpoints
      }
      Group {
        children IS humanoidBody
      }
    ]
  }
}

PROTO VisionSensor [
  exposedField SFVec3f    translation       0 0 0
  exposedField SFRotation rotation          0 0 1 0
  exposedField MFNode     children          [ ]
  exposedField SFFloat    fieldOfView       0.785398
  exposedField SFString   name              ""
  exposedField SFFloat    frontClipDistance 0.01
  exposedField SFFloat    backClipDistance  10.0
  exposedField SFString   type              "NONE"
  exposedField SFInt32    sensorId          -1
  exposedField SFInt32    width             320
  exposedField SFInt32    height            240
]
{
  Transform {
    rotation         IS rotation
    translation      IS translation
    children         IS children
  }
}
```

```
PROTO ForceSensor [
  exposedField SFVec3f    maxForce    -1 -1 -1
  exposedField SFVec3f    maxTorque   -1 -1 -1
  exposedField SFVec3f    translation 0 0 0
  exposedField SFRotation rotation    0 0 1 0
  exposedField SFInt32    sensorId    -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO Gyro [
  exposedField SFVec3f    maxAngularVelocity -1 -1 -1
  exposedField SFVec3f    translation        0 0 0
  exposedField SFRotation rotation           0 0 1 0
  exposedField SFInt32    sensorId           -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO AccelerationSensor [
  exposedField SFVec3f    maxAcceleration -1 -1 -1
  exposedField SFVec3f    translation     0 0 0
  exposedField SFRotation rotation        0 0 1 0
  exposedField SFInt32    sensorId        -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO PressureSensor [
  exposedField SFFloat    maxPressure -1
  exposedField SFVec3f    translation 0 0 0
  exposedField SFRotation rotation    0 0 1 0
  exposedField SFInt32    sensorId    -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO PhotoInterrupter [
  exposedField SFVec3f transmitter 0 0 0
  exposedField SFVec3f receiver    0 0 0
  exposedField SFInt32 sensorId    -1
]
{
  Transform{
    children [
```

```
      Transform{
        translation IS transmitter
      }
      Transform{
        translation IS receiver
      }
    ]
  }
}

PROTO CylinderSensorZ [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 1 0 0 1.5708
    children [
      DEF SensorY CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisY Transform{
        children [
          Transform{
            rotation 1 0 0 -1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorY.rotation_changed TO AxisY.set_rotation
}

PROTO CylinderSensorY [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 0 1 0 1.5708
    children [
      DEF SensorX CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisX Transform{
        children [
          Transform{
            rotation 0 1 0 -1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorX.rotation_changed TO AxisX.set_rotation
}
```

```
PROTO CylinderSensorX [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 0 0 1 -1.5708
    children [
      DEF SensorZ CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisZ Transform{
        children [
          Transform{
            rotation 0 0 1 1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorZ.rotation_changed TO AxisZ.set_rotation
}

NavigationInfo {
  avatarSize    0.5
  headlight     TRUE
  type  ["EXAMINE", "ANY"]
}

Background {
  skyColor 0.4 0.6 0.4
}

Viewpoint {
  position    3 0 0.835
  orientation 0.5770 0.5775 0.5775 2.0935
}

DEF box3 Humanoid {
  humanoidBody [
    DEF WAIST Joint {
      jointType "free"
      translation 0.55 -0.02 0.15
      rotation 0 1 0 0.2
      children [
        DEF BODY Segment {
        #====== wood block (cedar tree) ============
        mass 3
        momentsOfInertia [0.01 0 0 0 0.001 0 0 0 0.01]
        #====== aluminum block ==============
        #mass 86.9
        #momentsOfInertia [4.896 0 0 0 0.57599 0 0 0 4.896]
          children Transform {
          translation 0 0 0
          rotation 1 0 0 0
            children Shape {
              geometry Box {
```

```
                    size 0.25 0.6 0.1               }
                appearance Appearance {
                    material Material {
                        diffuseColor 1.0 1.0 0.0
                    }
                }
            }
        }
        }
      ]
    }
  ]
  joints [
    USE WAIST
  ]
  name "box3"
  segments [
    USE BODY
  ]
}
```

## 8.3 Colision_caja_sobre_brazos.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<grxui>
    <mode name="Simulation">
        <item class="com.generalrobotix.ui.item.GrxWorldStateItem"
name="untitled" select="true">
            <property name="logTimeStep" value="0.0050 "/>
            <property name="integrate" value="true"/>
            <property name="viewsimulate" value="false"/>
            <property name="totalTime" value="7.0 "/>
            <property name="timeStep" value="0.0050 "/>
            <property name="method" value="RUNGE_KUTTA"/>
            <property name="gravity" value="9.8 "/>
            <property name="viewsimulationTimeStep" value="0.033 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem"
name="floor" select="true" url="$(OPENHRPHOME)/etc/floor.wrl">
            <property name="isRobot" value="false"/>
            <property name="WAIST.rotation" value="0.0 1.0 0.0 0.0 "/>
            <property name="WAIST.translation" value="0.0 0.0 -0.1 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem"
name="box3" select="true" url="$(OPENHRPHOME)/etc/box3.wrl">
            <property name="isRobot" value="false"/>
            <property name="WAIST.rotation" value="0.0 0.0 0.0 0.0 "/>
            <property name="WAIST.translation" value="0.32 0.0 1.65
"/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem"
name="sample" select="true" url="$(OPENHRPHOME)/etc/sample.wrl">
            <property name="RLEG_HIP_R.angle" value="0.0 "/>
            <property name="RARM_SHOULDER_R.mode" value="HighGain"/>
            <property name="LLEG_KNEE.mode" value="HighGain"/>
            <property name="RARM_ELBOW.angle" value="-1.5708 "/>
            <property name="LLEG_ANKLE_P.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_P.angle" value="-0.0424675 "/>
            <property name="LLEG_ANKLE_R.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_R.angle" value="0.0 "/>
            <property name="LLEG_HIP_Y.mode" value="HighGain"/>
            <property name="RLEG_HIP_P.mode" value="HighGain"/>
            <property name="RARM_WRIST_P.angle" value="0.0 "/>
            <property name="CHEST.mode" value="HighGain"/>
            <property name="RARM_WRIST_R.angle" value="0.0 "/>
            <property name="RARM_WRIST_Y.angle" value="0.0 "/>
            <property name="RLEG_KNEE.angle" value="0.0785047 "/>
            <property name="RLEG_HIP_R.mode" value="HighGain"/>
            <property name="LARM_SHOULDER_P.angle" value="0.174533 "/>
            <property name="LARM_SHOULDER_R.angle" value="-0.00349066
"/>
            <property name="LARM_WRIST_P.mode" value="HighGain"/>
            <property name="LARM_SHOULDER_Y.angle" value="0.0 "/>
            <property name="LLEG_HIP_P.angle" value="-0.0360373 "/>
            <property name="LARM_WRIST_R.mode" value="HighGain"/>
            <property name="LLEG_HIP_R.angle" value="0.0 "/>
            <property name="WAIST.rotation" value="0.0 1.0 0.0 0.0 "/>
            <property name="LLEG_HIP_Y.angle" value="0.0 "/>
            <property name="LARM_ELBOW.angle" value="-1.5708 "/>
            <property name="LARM_SHOULDER_Y.mode" value="HighGain"/>
```

```
            <property name="setupDirectory" value="$
(OPENHRPHOME)/Controller/rtc/SampleHG"/>
            <property name="RLEG_KNEE.mode" value="HighGain"/>
            <property name="CHEST.angle" value="0.0 "/>
            <property name="WAIST_P.mode" value="HighGain"/>
            <property name="LLEG_HIP_P.mode" value="HighGain"/>
            <property name="LLEG_ANKLE_P.angle" value="-0.0424675 "/>
            <property name="RARM_SHOULDER_P.angle" value="0.174533 "/>
            <property name="LLEG_ANKLE_R.angle" value="0.0 "/>
            <property name="RARM_SHOULDER_R.angle" value="-0.00349066
"/>
            <property name="WAIST_R.mode" value="HighGain"/>
            <property name="LLEG_KNEE.angle" value="0.0785047 "/>
            <property name="LLEG_HIP_R.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_Y.angle" value="0.0 "/>
            <property name="LARM_WRIST_P.angle" value="0.0 "/>
            <property name="LARM_WRIST_R.angle" value="0.0 "/>
            <property name="LARM_WRIST_Y.angle" value="0.0 "/>
            <property name="RARM_WRIST_Y.mode" value="HighGain"/>
            <property name="RARM_ELBOW.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_Y.mode" value="HighGain"/>
            <property name="CHEST_P.angle" value="0.0 "/>
            <property name="WAIST.mode" value="HighGain"/>
            <property name="WAIST_P.angle" value="0.0 "/>
            <property name="LARM_SHOULDER_P.mode" value="HighGain"/>
            <property name="WAIST_R.angle" value="0.0 "/>
            <property name="CHEST_Y.angle" value="0.0 "/>
            <property name="WAIST_Y.angle" value="0.0 "/>
            <property name="LARM_SHOULDER_R.mode" value="HighGain"/>
            <property name="WAIST.translation" value="0.0 0.0 0.77975
"/>
            <property name="RLEG_HIP_Y.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_P.mode" value="HighGain"/>
            <property name="LARM_WRIST_Y.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_R.mode" value="HighGain"/>
            <property name="controller" value="SampleHGController"/>
            <property name="RARM_WRIST_P.mode" value="HighGain"/>
            <property name="isRobot" value="true"/>
            <property name="LARM_ELBOW.mode" value="HighGain"/>
            <property name="controlTime" value="0.002"/>
            <property name="setupCommand" value="SampleHG$(BIN_SFX)"/>
            <property name="RARM_WRIST_R.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_P.mode" value="HighGain"/>
            <property name="RLEG_HIP_P.angle" value="-0.0360373 "/>
            <property name="RLEG_HIP_Y.angle" value="0.0 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxCollisionPairItem"
name="CP#floor#sample" select="true">
            <property name="springConstant" value="0 0 0 0 0 0"/>
            <property name="slidingFriction" value="0.5"/>
            <property name="jointName2" value=""/>
            <property name="jointName1" value=""/>
            <property name="sprintDamperModel" value="false"/>
            <property name="damperConstant" value="0 0 0 0 0 0"/>
            <property name="objectName2" value="sample"/>
            <property name="objectName1" value="floor"/>
            <property name="staticFriction" value="0.5"/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxCollisionPairItem"
name="CP#box3#sample" select="true">
            <property name="springConstant" value="0 0 0 0 0 0"/>
```

```
            <property name="slidingFriction" value="0.5"/>
            <property name="jointName2" value=""/>
            <property name="jointName1" value=""/>
            <property name="sprintDamperModel" value="false"/>
            <property name="damperConstant" value="0 0 0 0 0 0"/>
            <property name="objectName2" value="sample"/>
            <property name="objectName1" value="box3"/>
            <property name="staticFriction" value="0.5"/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxCollisionPairItem"
name="CP#floor#box3" select="true">
            <property name="springConstant" value="0 0 0 0 0 0"/>
            <property name="slidingFriction" value="0.5"/>
            <property name="jointName2" value=""/>
            <property name="jointName1" value=""/>
            <property name="damperConstant" value="0 0 0 0 0 0"/>
            <property name="objectName2" value="box3"/>
            <property name="objectName1" value="floor"/>
            <property name="springDamperModel" value="false"/>
            <property name="staticFriction" value="0.5"/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxGraphItem"
name="GraphList1" select="true">
            <property name="Graph1.dataItems" value=""/>
            <property name="Graph2.dataItems" value=""/>
            <property name="Graph0.dataItems" value=""/>
            <property name="Graph3.dataItems" value=""/>
        </item>

        <windowconfig>
            <window fullScreen="false" height="874" root="true"
width="1448" x="-4" y="-4">
                <layout position="Center" splitratio="0.601">
                    <layout position="Center" splitratio="0.314">
                        <layout position="Center" splitratio="0.313">
                            <layout position="Center"
splitratio="0.0">
                                <tab name="Jython Prompt"
select="true"/>
                                <tab name="NameService Monitor"/>
                                <tab name="Process Manager"/>
                            </layout>
                            <layout position="West" splitratio="0.0">
                                <tab name="Item View" select="true"/>
                            </layout>
                        </layout>
                        <layout position="South" splitratio="0.0">
                            <tab name="3DView" select="true"/>
                            <tab name="OpenHRP"/>
                            <tab name="Text Editor"/>
                        </layout>
                    </layout>
                    <layout position="East" splitratio="0.0">
                        <tab name="Graph"/>
                        <tab name="Robot State" select="true"/>
                        <tab name="Property"/>
                    </layout>
                </layout>
            </window>
        </windowconfig>
    </mode></grxui>
```

## 8.4 SampleHG.cpp

```
// -*- mode: c++; indent-tabs-mode: t; tab-width: 4; c-basic-offset:
4; -*-
/*
 * Copyright (c) 2008, AIST, the University of Tokyo and General
Robotix Inc.
 * All rights reserved. This program is made available under the terms
of the
 * Eclipse Public License v1.0 which accompanies this distribution,
and is
 * available at http://www.eclipse.org/legal/epl-v10.html
 * Contributors:
 * National Institute of Advanced Industrial Science and Technology
(AIST)
 * General Robotix Inc.
 */
/*!
 * @file  SampleHG.cpp
 * @brief Sample LF component
 * $Date$
 *
 * $Id$
 */

#include "SampleHG.h"

#include <iostream>

#define DOF (28)

#define ANGLE_FILE "etc/angle.dat"
#define VEL_FILE   "etc/vel.dat"
#define ACC_FILE   "etc/acc.dat"

namespace {
  const bool CONTROLLER_BRIDGE_DEBUG = false;
}


// Module specification
// <rtc-template block="module_spec">
static const char* samplepd_spec[] =
  {
    "implementation_id", "SampleHG",
    "type_name",         "SampleHG",
    "description",       "Sample HG component",
    "version",           "0.1",
    "vendor",            "AIST",
    "category",          "Generic",
    "activity_type",     "DataFlowComponent",
    "max_instance",      "10",
    "language",          "C++",
    "lang_type",         "compile",
    // Configuration variables

    ""
  };
// </rtc-template>
```

```
SampleHG::SampleHG(RTC::Manager* manager)
  : RTC::DataFlowComponentBase(manager),
    // <rtc-template block="initializer">
    m_angleOut("angle", m_angle),
    m_velOut("vel", m_vel),
    m_accOut("acc", m_acc)

    // </rtc-template>
{
  if( CONTROLLER_BRIDGE_DEBUG )
  {
    std::cout << "SampleHG::SampleHG" << std::endl;
  }
  // Registration: InPort/OutPort/Service
  // <rtc-template block="registration">
  // Set InPort buffers

  // Set OutPort buffer
  registerOutPort("angle", m_angleOut);
  registerOutPort("vel", m_velOut);
  registerOutPort("acc", m_accOut);
  // Set service provider to Ports

  // Set service consumers to Ports

  // Set CORBA Service Ports

  // </rtc-template>

  if (access(ANGLE_FILE, 0)){
    std::cerr << ANGLE_FILE << " not found" << std::endl;
  }else{
    angle.open(ANGLE_FILE);
  }

  if (access(VEL_FILE, 0)){
    std::cerr << VEL_FILE << " not found" << std::endl;
  }else{
    vel.open(VEL_FILE);
  }

  if (access(ACC_FILE, 0))
  {
    std::cerr << ACC_FILE << " not found" << std::endl;
  }else{
    acc.open(ACC_FILE);
  }

  m_angle.data.length(DOF);
  m_vel.data.length(DOF);
  m_acc.data.length(DOF);

}

SampleHG::~SampleHG()
{
      if (angle.is_open())  angle.close();
      if (vel.is_open())    vel.close();
      if (acc.is_open())    acc.close();
}
```

```
RTC::ReturnCode_t SampleHG::onInitialize()
{
  // <rtc-template block="bind_config">
  // Bind variables and configuration variable
  if( CONTROLLER_BRIDGE_DEBUG )
  {
    std::cout << "onInitialize" << std::endl;
  }

  // </rtc-template>
  return RTC::RTC_OK;
}



/*
RTC::ReturnCode_t SampleHG::onFinalize()
{
  return RTC::RTC_OK;
}
*/

/*
RTC::ReturnCode_t SampleHG::onStartup(RTC::UniqueId ec_id)
{
  return RTC::RTC_OK;
}
*/

/*
RTC::ReturnCode_t SampleHG::onShutdown(RTC::UniqueId ec_id)
{
  return RTC::RTC_OK;
}
*/

RTC::ReturnCode_t SampleHG::onActivated(RTC::UniqueId ec_id)
{

      std::cout << "on Activated" << std::endl;
      angle.seekg(0);
      vel.seekg(0);
      acc.seekg(0);

      return RTC::RTC_OK;
}

/*
RTC::ReturnCode_t SampleHG::onDeactivated(RTC::UniqueId ec_id)
{
  return RTC::RTC_OK;
}
*/


RTC::ReturnCode_t SampleHG::onExecute(RTC::UniqueId ec_id)
{
  if( CONTROLLER_BRIDGE_DEBUG )
  {
    std::cout << "SampleHG::onExecute" << std::endl;
```

```
  }
  // ,±,ÌŠÖ”,Ì‚U,é•‘,¢,ÍController_impl::control,Ì”h‚¶‚æ‰¼‘zŠÖ”,É‘Î
‰ž,·,é
  double dummy;
  angle >> dummy; vel >> dummy; acc >> dummy; // skip time
  int i;

  //Šeƒtƒ@ƒCƒ‹,©,çƒfˆ[ƒ^,ðˆê‚s“Ç,Ý‚ž,ñ,Åƒ|ˆ[ƒg,É—¬,·
  for (i=0; i<DOF; i++)
  {
      angle >> m_angle.data[i];
      vel   >> m_vel.data[i];
      acc   >> m_acc.data[i];
  }

  m_angleOut.write();
  m_velOut.write();
  m_accOut.write();

  return RTC::RTC_OK;
}


/*
  RTC::ReturnCode_t SampleHG::onAborting(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onError(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onReset(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onStateUpdate(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onRateChanged(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/



extern "C"
```

```
{

	DllExport void SampleHGInit(RTC::Manager* manager)
	{
		RTC::Properties profile(samplepd_spec);
		manager->registerFactory(profile,
                                                 RTC::Create<SampleHG>,
RTC::Delete<SampleHG>);
	}

};
```